Nutzung von NetBeans

HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

Thema:	Nutzung von NetBeans
Autor:	Prof. Dr. Stephan Kleuker
Version/Datum:	4.2.1 / 14.12.2018
Empfänger:	Teilnehmer der Lehrveranstaltungen Komponentenbasierte Software-Entwicklung / Datenbanken / Programmierung 1 / Objektorientierte Analyse und Design / Qualitätssicherung

Die folgende Einführung bietet eine Einstiegsmöglichkeit in die Nutzung von NetBeans. Man beachte, dass diese Einführung nicht kontinuierlich aktualisiert wird, trotzdem sind Nachfragen und Ergänzungswünsche sehr willkommen. Dieser Text soll, kann und wird kein Skript bzw. eine intensive Einarbeitung in die teilweise komplexen Themen ersetzen, da ausschließlich Bedienungshinweise, aber fast keine Hintergründe vorgestellt werden. Die formale Form entspricht einem kontinuierlich wachsenden internen Manual und erfüllt damit bei Weitem keine Anforderungen an die Grammatik, Rechtschreibung, Zeichensetzung und weitere formale Aspekte, wie die korrekte Einbindung von Abbildungen in den Text. Hinweise auf Fehler werden trotzdem gerne angenommen.

Der folgende Text ist immer so strukturiert, dass zunächst beschrieben wird, was gemacht werden soll und danach das zugehörige Bildschirmfoto angegeben ist.

Die ersten Versionen dieser Datei bezogen sich auf NetBeans 6.7.1. Seitdem sind einige Details für NetBeans 6.8, 6.9.1, 7.0.1, 7.2, 7.3.1, 8.0, 8.0.2, 8.1, 8.2 aktualisiert worden, wobei erwartet wird, das zentrale Aussagen auch auf aktuellere NetBeans-Versionen übertragen werden können. Hinweise auf wesentliche, hier nicht behandelte Änderungen nimmt der Autor gerne entgegen. Durch die kontinuierliche Bearbeitung wurden stammen Bilder teilweise aus Englisch- bzw. Deutsch-sprachigen Versionen. Da sich die Oberflächen aber inhaltlich nicht ändern, sollte dies keine Probleme bereiten.

Inhaltsverzeichnis

1 Installation von Java und Java DB 5
1.1 Installation von Java 5
1.2 Setzen von Systemvariablen 17
1.3 Installation von Apache Derby (früher Java DB) 33
1.4 Überprüfung der Installation 36
2 Installation von NetBeans (NetBeans 8) 39
2.1 Standardinstallation 39
2.2 Installation eines eigenen GlassFish-Servers 57
2.3 Einbindung eines neuen GlassFish-Servers in NetBeans 58
3 Erste Schritte mit NetBeans (NetBeans 8) 66
3.1 Erster Start 66
3.2 Anlegen eines Java-Projekts 69
3.3 Ausführen eines Programms 72
3.4 Mehrere ausführbare Programme 73
3.5 Anlegen einer neuen Klasse 76
3.6 Syntaxprüfung und Fehlerbearbeitung 78
3.7 Zeilennummern 79
3.8 Umbenennung von Projekten 79

3.9 Portables NetBeans 80
4 Hilfreiche NetBeans-Funktionalität im Java-Alltag 84
4.1 Code-Vervollständigung 84
4.2 Generierung von Gettern und Settern 85
4.3 Umbenennungen / Refactoring 87
4.4 Kommentierung 91
4.5 Organisation von Imports 91
4.6 Nutzung von Bibliotheken/Jar-Files (NetBeans 8) 93
4.7 Installation einer neueren/weiteren Java-Version 100
5 Nutzung von JUnit (NetBeans 8) 110
5.1 Vollständige Nutzung der JUnit-NetBeans-Unterstützung 110
5.2 Erstellung einer neuen Test-Klasse 119
5.3 Einblendung von „Test Packages“ 121
5.4 Direkte Nutzung der JUnit-Library von NetBeans 123
5.5 Nutzung eigener Test-Bibliotheken 125
6 NetBeans und Datenbanken (NetBeans 8) 128
6.1 Integration von Apache Derby /Java DB in NetBeans 128
6.2 Anlegen einer neuen Datenbank 130
6.3 Anlegen einer Tabelle 134
6.4 Speichern und Laden von SQL-Skripten 137
6.5 Ausführung von SQL-Befehlen 143
6.6 Zugriff mit JDBC 147
6.7 Einbindung existierender Java DB-Datenbanken 151
6.8 Stored Procedures und Trigger 158
6.9 AccessControlException beim Einspielen einer jar-Datei 167
6.10 Zugriffsprobleme durch verschiedene Betriebssysteme oder Nutzer 173
7 JPA (NetBeans 8) 177
7.1 Vorbereitung der JPA-Nutzung 177
7.2 Neue Entitys anlegen 180
7.3 Nutzung eines EntityManagers 186
7.4 Umwandlung von Klassen in JPA-Entity-Klassen 188
8 JSP (NetBeans 6.7) 194
8.1 Erstellung eines JSP-Projekts 194
8.2 Deployment und Ausführung eines einfachen Projekts 197
8.3 Fehler in JSPs korrigieren 203
8.4 Informationen zum GlassFish-Server 208
8.5 Nutzereinrichtung in GlassFish 213
8.6 Sichere JSP-Seiten 222
8.7 HTTPS-Sicherheitseinstellungen in Browsern für JSP und JSF 235
8.8 Probleme mit Fehlerseiten 252
8.9 Erstellung einer eigenen Tag Library 255
8.10 JSP mit Datenbanken 267
9 Servlets (NetBeans 6.7) 273
9.1 Anlegen eines Servlets 273
9.2 Ausführen eines Servlets 278
9.3 Minimales Servlet-Projekt 280
10 JSF (NetBeans 8) 287

10.1 Anpassen der JDK-Version 287
10.2 Erstellung eines JSF-Projekts 294
10.3 Entwicklung eines Minimalprojekts 300
10.4 Einbindung von JPA mit EJB-Hilfe 313
10.4.1 Einrichten der Persistence Unit 314
10.4.2 Anlegen der beans.xml 319
10.4.3 Erstellung der Entity-Klasse 321
10.4.4 Anlegen der EJB 323
10.4.5 Einbindung der EJB in der ManagedBean mit Injektion 326
10.5 Check-Liste bei Fehlern 332
10.6 Testen von JEE-Programmen mit Arquillian 334
10.6.1 Herunterladen vom Hersteller 335
10.6.2 Herunterladen der von Maven genutzten Version 339
10.6.3 Embedded Glassfish 344
10.6.4 Beispielnutzung 345
10.7 Sichere JSF-Applikationen 351
10.7.1 Einrichten der Berechtigungstabellen 352
10.7.2 Datenbank in NetBeans bekannt machen 353
10.7.3 Datenbank über JNDI zugängig machen 358
10.7.4 Eigenen Realm anlegen 360
10.7.5 GlassFish-spezifische Einstellungen 364
10.7.6 Einstellungen in JSF 368
10.8 Behandlung mit Fehlerseiten 381
10.9 Anmerkungen zu CDI in NetBeans 8.0.1 385
10.10 Nutzung von RichFaces 4.0.0 für JSF 2.0 (NetBeans 6.9) 386
10.11 Nutzung von RichFaces 3.3.2 für JSF 1.2 (NetBeans 6.7) 391
10.12 Installation eines neuen Servers 394
11 JavaFX (NetBeans 8.1) 401
11.1 Aktualisierung von NetBeans 401
11.2 Erstellung eines JavaFX-Projekts 403
11.3 JavaFX und NetBeans 8.0.2 412
11.4 Installation des Scene Builders 413
12 Selenium 424
12.1 Selenium IDE 424
12.2 Selenium WebDriver 433
12.3 Selenium WebDriver mit portabler Browser-Installation 443
13 Einführung in den Profiler 449
14 Tipps und Fragen 455
14.1 Sehr langsame Übersetzung von Web-Projekten 455
14.2 Meldung „BUILD FAILED“ 458
15 Installation von PlugIns - MongoDB Client 467
15.1 Installation eines NetBeans-Plugins am Beispiel NBMongo 468
15.2 Aufbau einer Datenbank-Verbindung 476
15.3 Erste Nutzung des MongoDB-Clients 479
15.4 Installation eines heruntergeladenen Plugins - NetBeans Command Shortcuts 487
15.5 Nutzung von NetBeans Command Shortcuts 490
16 MQTT 498

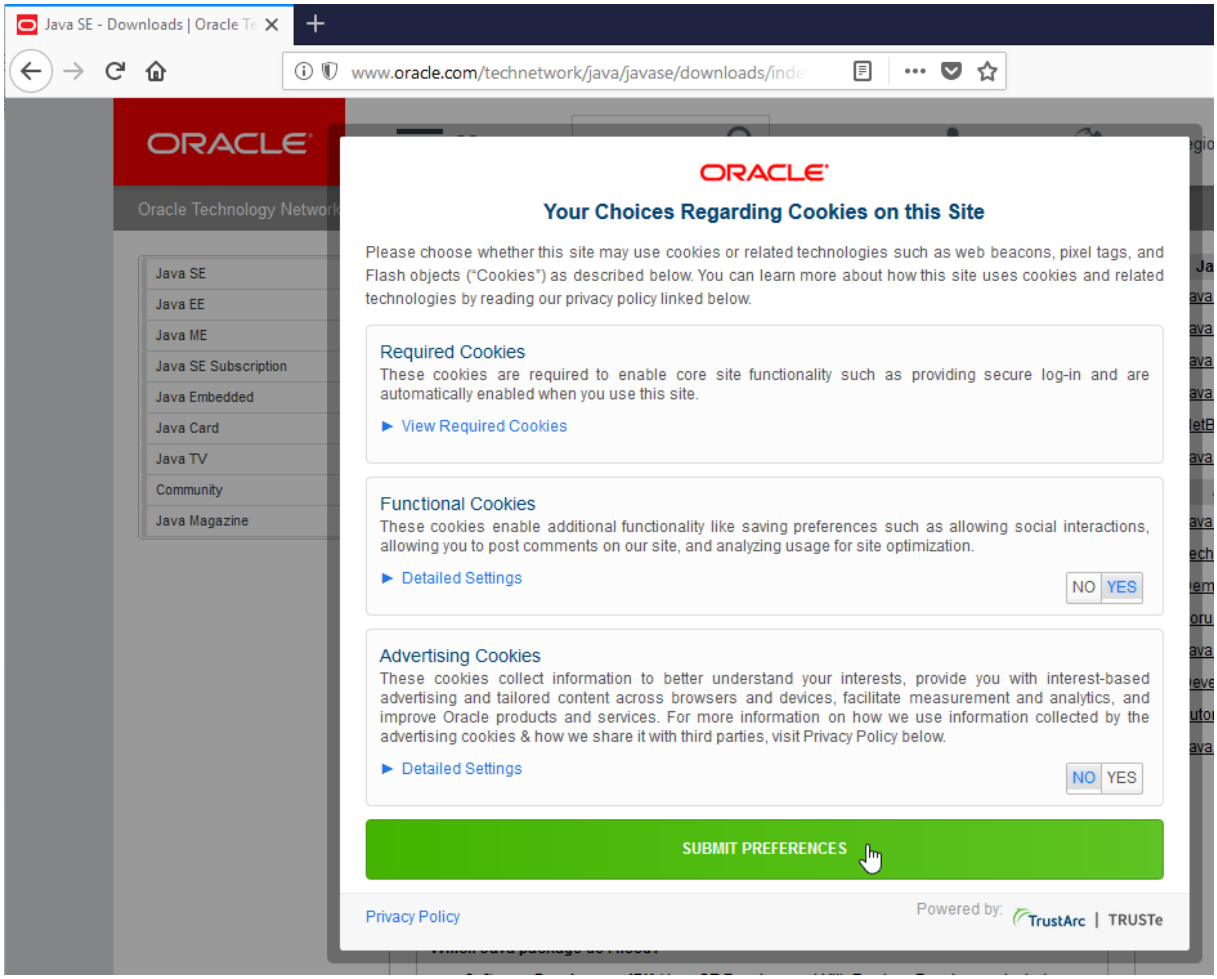
16.1 Installation Apache Apollo 498
16.2 Client-Entwicklung mit Eclipse paho 507
17 MongoDB 513
17.1 Installation 513
17.2 Start der Datenbank 516
17.3 Erste Nutzung 520
17.4 Java und MongoDB 521
18 UMLet 529
18.1 Installation 529
18.2 Erste Nutzung 531
18.3 Verknüpfung von uxf-Dateien mit UMLet 542
18.4 Anlegen einer UMLet-Datei in NetBeans 544
18.5 Öffnen einer UMLet-Datei in NetBeans 547
18.6 Öffnen einer UMLet-Datei in NetBeans bei fehlendem Menü-Punkt 547
18.7 Einbindung von UMLet in NetBeans mit Schaltfläche 548
18.8 Nutzung von UMLet in NetBeans 551
19 Überdeckungstests mit JaCoCo/TikiOne 555
20 Metriken mit Simple Code Metrics 570
21 SQLChecker 576
22 SQLCheckerCreator 583

1 Installation von Java und Java DB

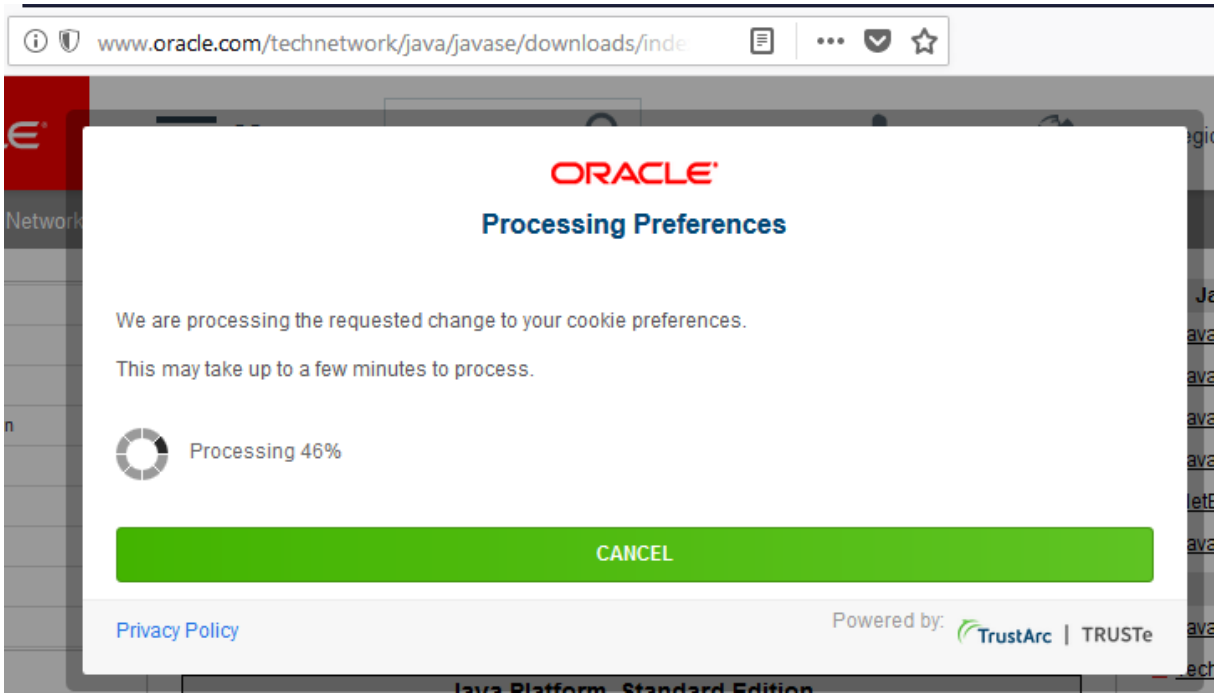
Vorbemerkung: Wie alle Programmiersprachen entwickelt sich Java weiter, dabei werden meist neue Programmbibliotheken ergänzt oder hinzugefügt, die Funktionalität anbieten, die bisher in Java noch nicht vorhanden war. Weiterhin werden Befehle ergänzt, die es erfahrenen Programmierern erleichtern, etwas kürze Programme zu schreiben. Bis einschließlich der Version 8 liefen (praktisch) alle in anderen Versionen von Java geschriebenen Programme auch in dieser Version. Dies ist ab der Version 9 nicht mehr ganz der Fall, was einige Vorteile (modularere Software-Entwicklung, Auslieferung ohne vollständige Java Run Time) hat, aber langfristig kritische strategische Auswirkungen haben kann. Andere Programmiersprachen wie PHP und einige JavaScript-Frameworks haben auch massiver gegen diese Abwärtskompatibilität verstoßen und nicht wesentlich in ihrer Beliebtheit eingebüßt. Trotzdem ist zu erwarten, dass viele Projekte auch weit über 2018 hinaus auf Java 8 basieren, weshalb es Grundlage dieser Installation ist. Weiterhin bleibt anzumerken, dass bis Java 8 zwischen den Wechsel der Hauptversionsnummer, z. B. zwischen Java 7 und Java 8 oft mehrere Jahre lagen, nach einer neuen Release-Strategie diese Nummer aber durchaus mehrfach im Jahr verändert werden kann. Für Programmieranfänger sei angemerkt, dass alles Lehrmaterial was ab Java 7 (teilweise Java 5) entstanden ist relativ problemlos weiter genutzt werden kann.

1.1 Installation von Java

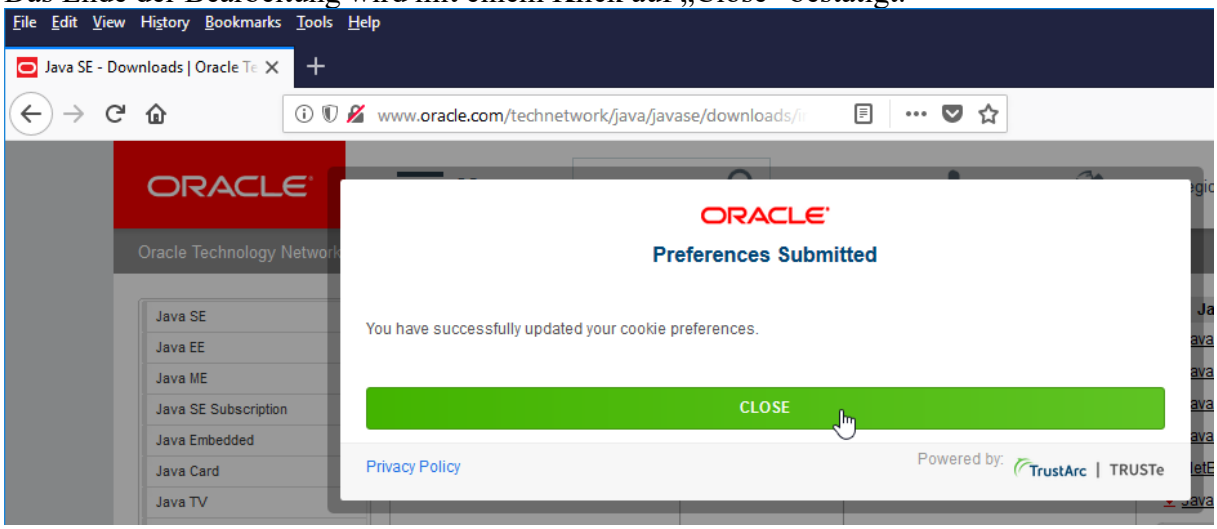
Java wird direkt von der Oracle-Seite http://www.oracle.com/technetwork/java/javase/downloads/index.html heruntergeladen, dabei wird das Development Kit (JDK), auch „Java for Developers“ genannt, und nicht nur die Runtime-Version, benötigt. Da sich die Startseite kontinuierlich ändert, kann hier nur eine Version, September 2018, berücksichtigt werden. In diesem Fall müssen abhängig vom Browser zuerst zwei Cookie-Regeln mit NO oder YES festgelegt werden. Die Eingabe endet mit einem Klick auf „Submit Preferences“.



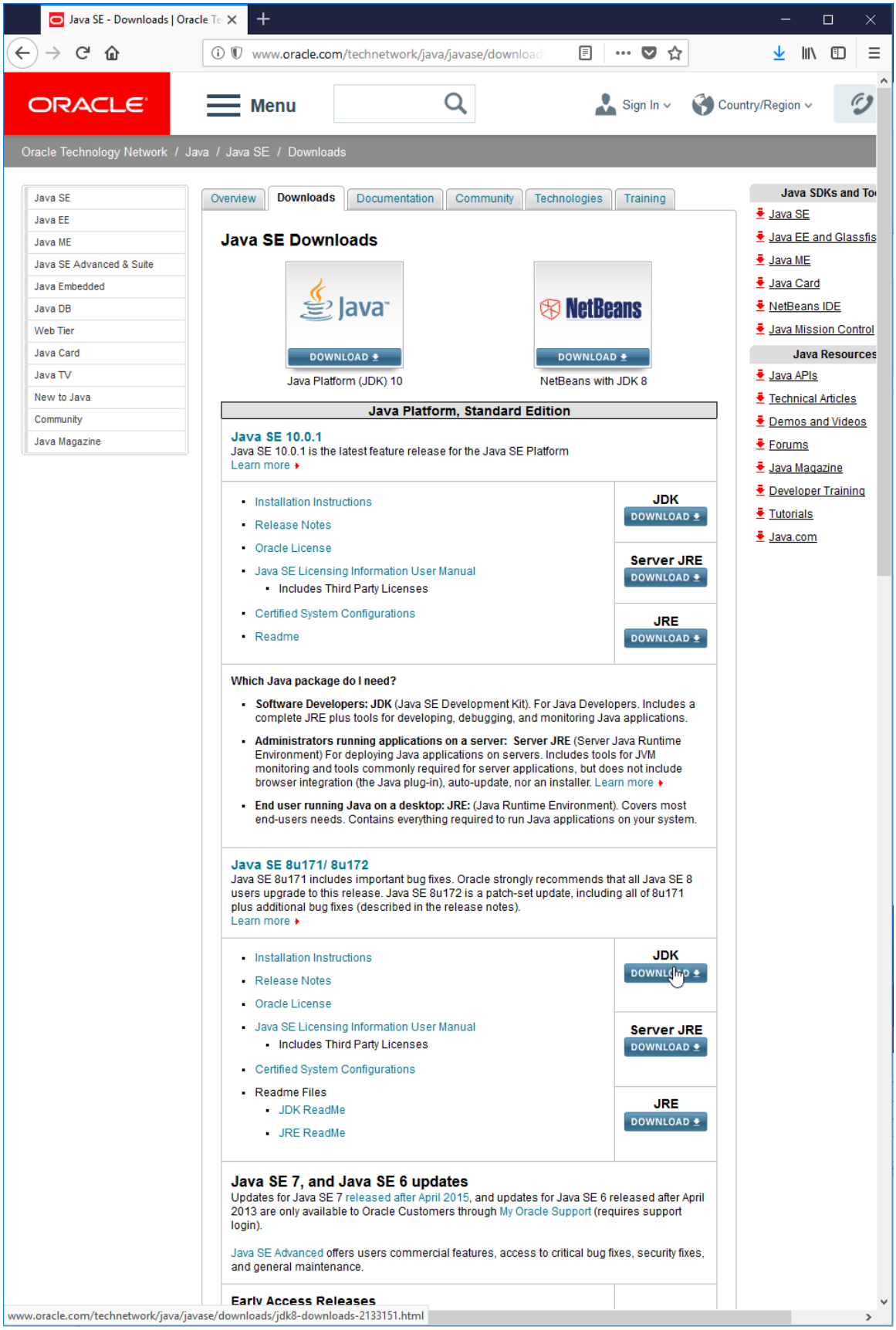
Die Bearbeitung kann etwas dauern.



Das Ende der Bearbeitung wird mit einem Klick auf „Close“ bestätigt.

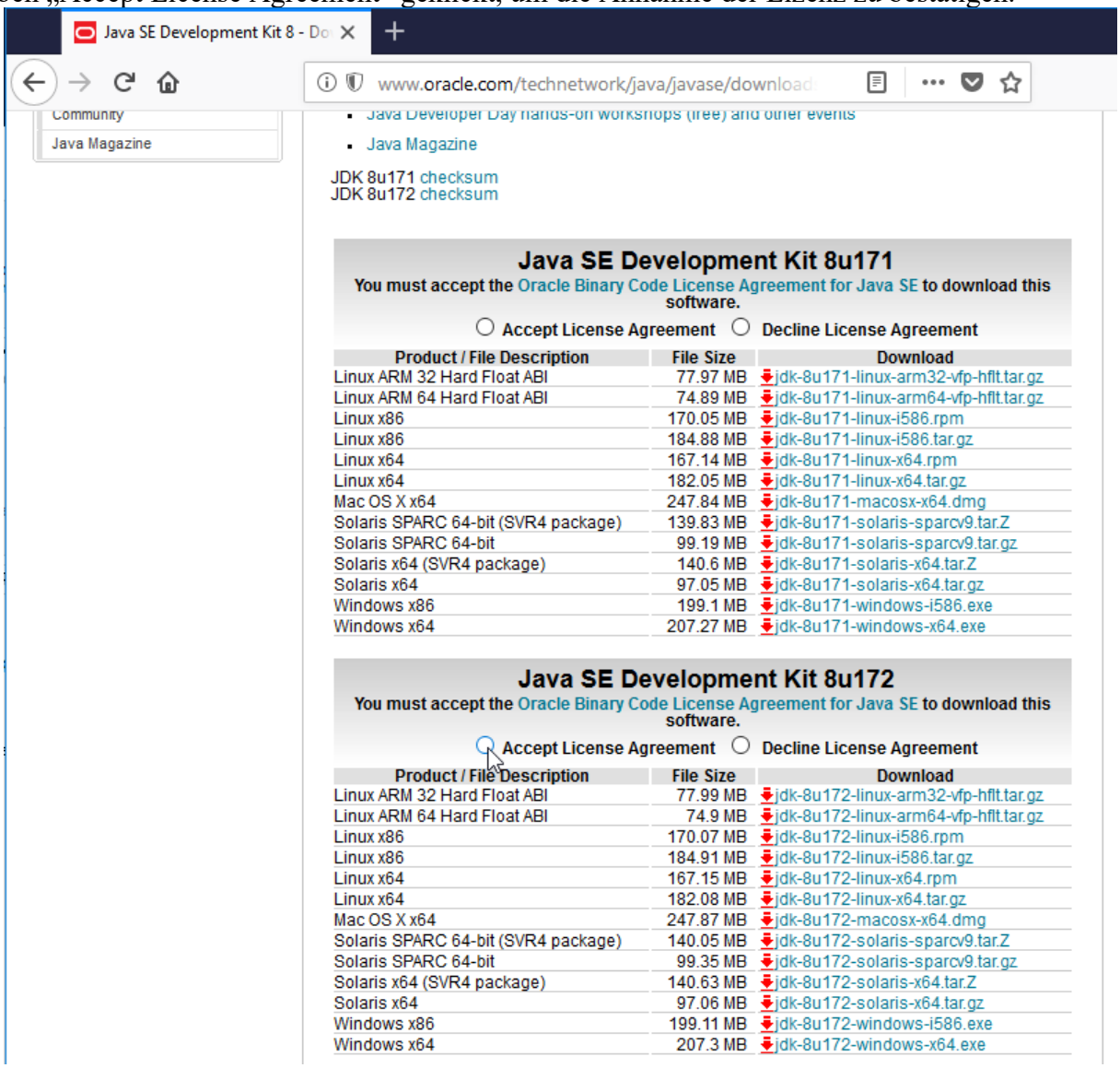


Im ersten Schritt wird zunächst die Java SE-Variante für die Standard Edition ausgewählt. Die anderen Varianten bieten keine Vorteile für Anfänger und enthalten teilweise nicht alle Bestandteile der Standard Edition.
Die genaue Java-Version, sogar ob es Version 6.x oder Version 7 ist, spielt fast keine Rolle, die Hauptversionsnummer sollte unter 9 liegen, hier wird Java SE 8u172 ausgewählt und unter „JDK“ auf „Download“ geklickt.

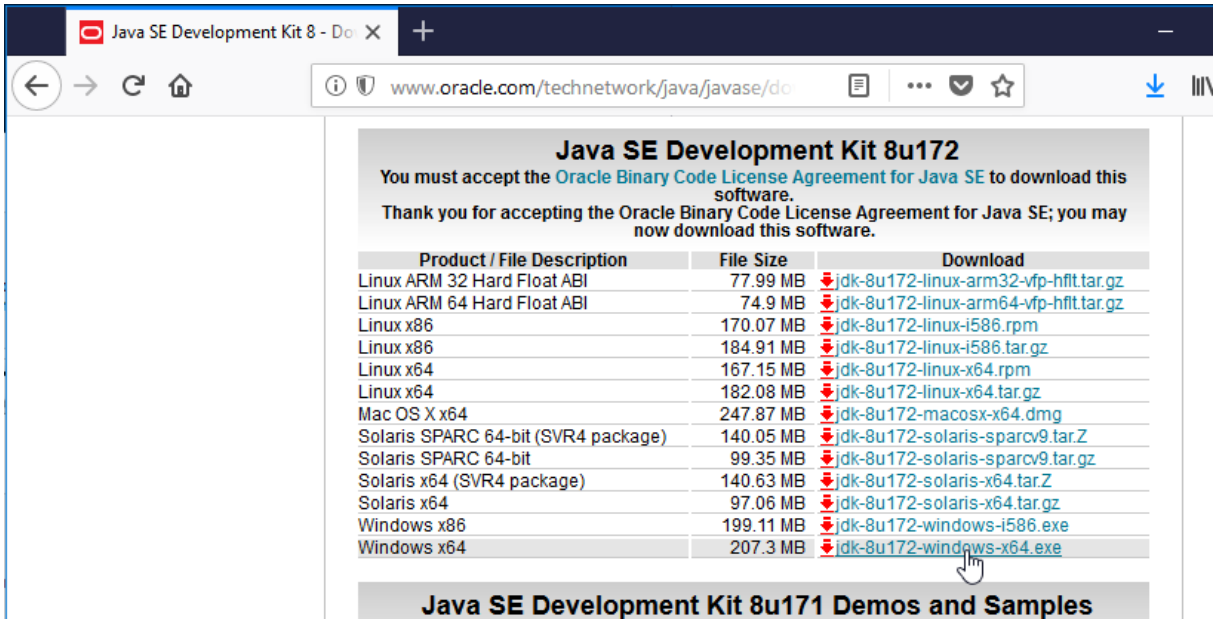

Java SE - Downloads | Oracle Te
www.oracle.com/technetwork/java/javase/downloa
ORACLE
Menu
Sign In
Country/Region
Oracle Technology Network / Java / Java SE / Downloads
Java SE
Java EE
Java ME
Java SE Advanced & Suite
Java Embedded
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine
Overview
Downloads
Documentation
Community
Technologies
Training
Java SDKs and To
Java SE
Java EE and Glassfis
Java ME
Java Card
NetBeans IDE
Java Mission Control
Java Resources
Java APIs
Technical Articles
Demos and Videos
Forums
Java Magazine
Developer Training
Tutorials
Java.com
Java SE Downloads
Java
DOWNLOAD
Java Platform (JDK) 10
NetBeans
DOWNLOAD
NetBeans with JDK 8
Java Platform, Standard Edition
Java SE 10.0.1
Java SE 10.0.1 is the latest feature release for the Java SE Platform
Learn more
Installation Instructions
Release Notes
Oracle License
Java SE Licensing Information User Manual
Includes Third Party Licenses
Certified System Configurations
Readme
JDK
DOWNLOAD
Server JRE
DOWNLOAD
JRE
DOWNLOAD
Which Java package do I need?
Software Developers: JDK (Java SE Development Kit). For Java Developers. Includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.
Administrators running applications on a server: Server JRE (Server Java Runtime Environment) For deploying Java applications on servers. Includes tools for JVM monitoring and tools commonly required for server applications, but does not include browser integration (the Java plug-in), auto-update, nor an installer. Learn more
End user running Java on a desktop: JRE: (Java Runtime Environment). Covers most end-users needs. Contains everything required to run Java applications on your system.
Java SE 8u171/ 8u172
Java SE 8u171 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release. Java SE 8u172 is a patch-set update, including all of 8u171 plus additional bug fixes (described in the release notes).
Learn more
Installation Instructions
Release Notes
Oracle License
Java SE Licensing Information User Manual
Includes Third Party Licenses
Certified System Configurations
Readme Files
JDK ReadMe
JRE ReadMe
JDK
DOWNLOAD
Server JRE
DOWNLOAD
JRE
DOWNLOAD
Java SE 7, and Java SE 6 updates
Updates for Java SE 7 released after April 2015, and updates for Java SE 6 released after April 2013 are only available to Oracle Customers through My Oracle Support (requires support login).
Java SE Advanced offers users commercial features, access to critical bug fixes, security fixes, and general maintenance.
Early Access Releases
www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Wichtig ist, dass ein JDK, ein Java Development Kit, heruntergeladen wird, da nur damit die eigene Entwicklung möglich ist. Mit einer JRE, einer Java Runtime Environment, können nur fertige Java-Programme ausgeführt werden.

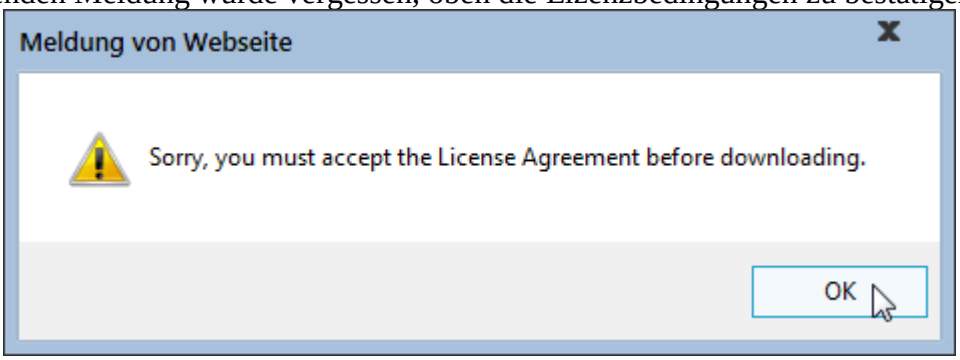
Zunächst werden die Lizenzbedingungen (Link im mittleren Kasten) gelesen und dann der Kreis links neben „Accept License Agreement“ geklickt, um die Annahme der Lizenz zu bestätigen.



Passend zum eigenen Betriebssystem (hier die Beschreibung für Windows 64 Bit, meist mit x64 bezeichnet; für Windows 32 Bit ist es x86) und klickt den Link unter „Download“.



Bei der folgenden Meldung wurde vergessen, oben die Lizenzbedingungen zu bestätigen.

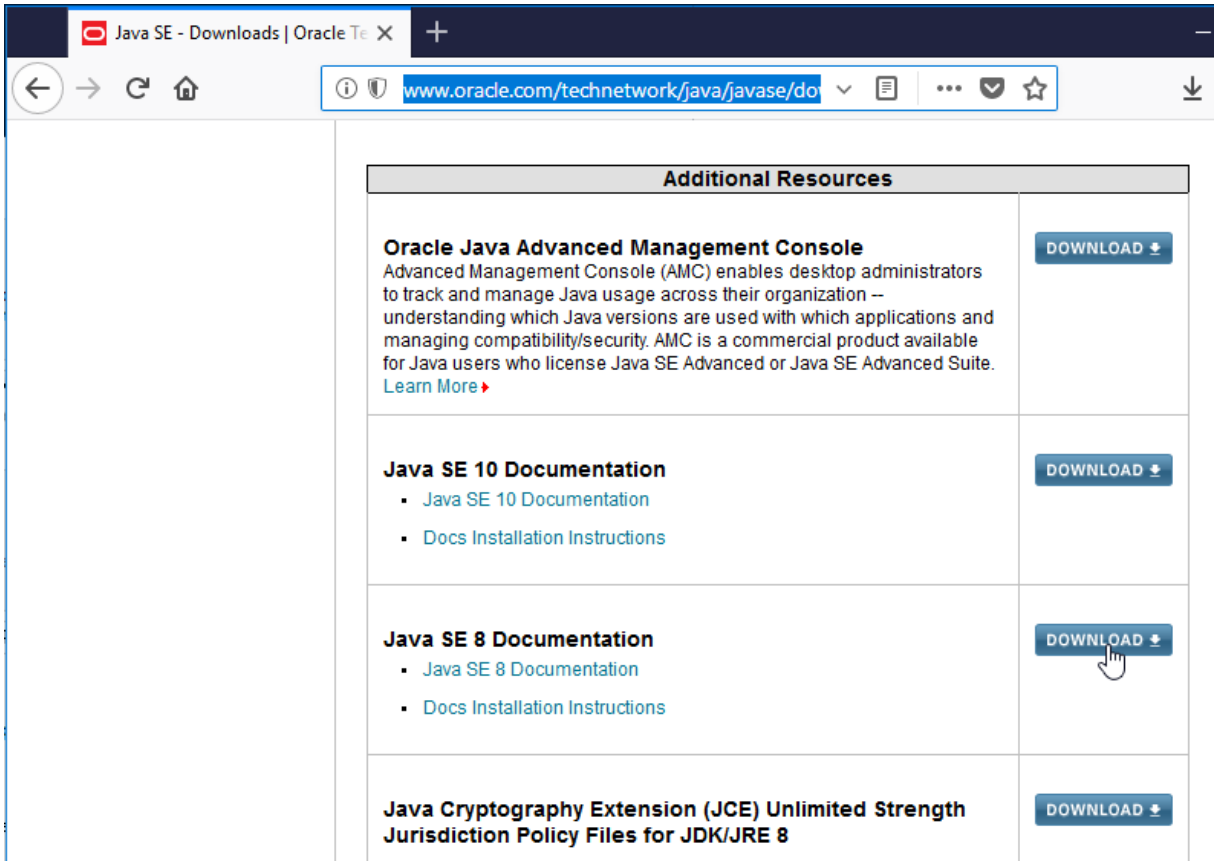


Dann kann der Download durch einen Klick auf „Save File“ stattfinden.

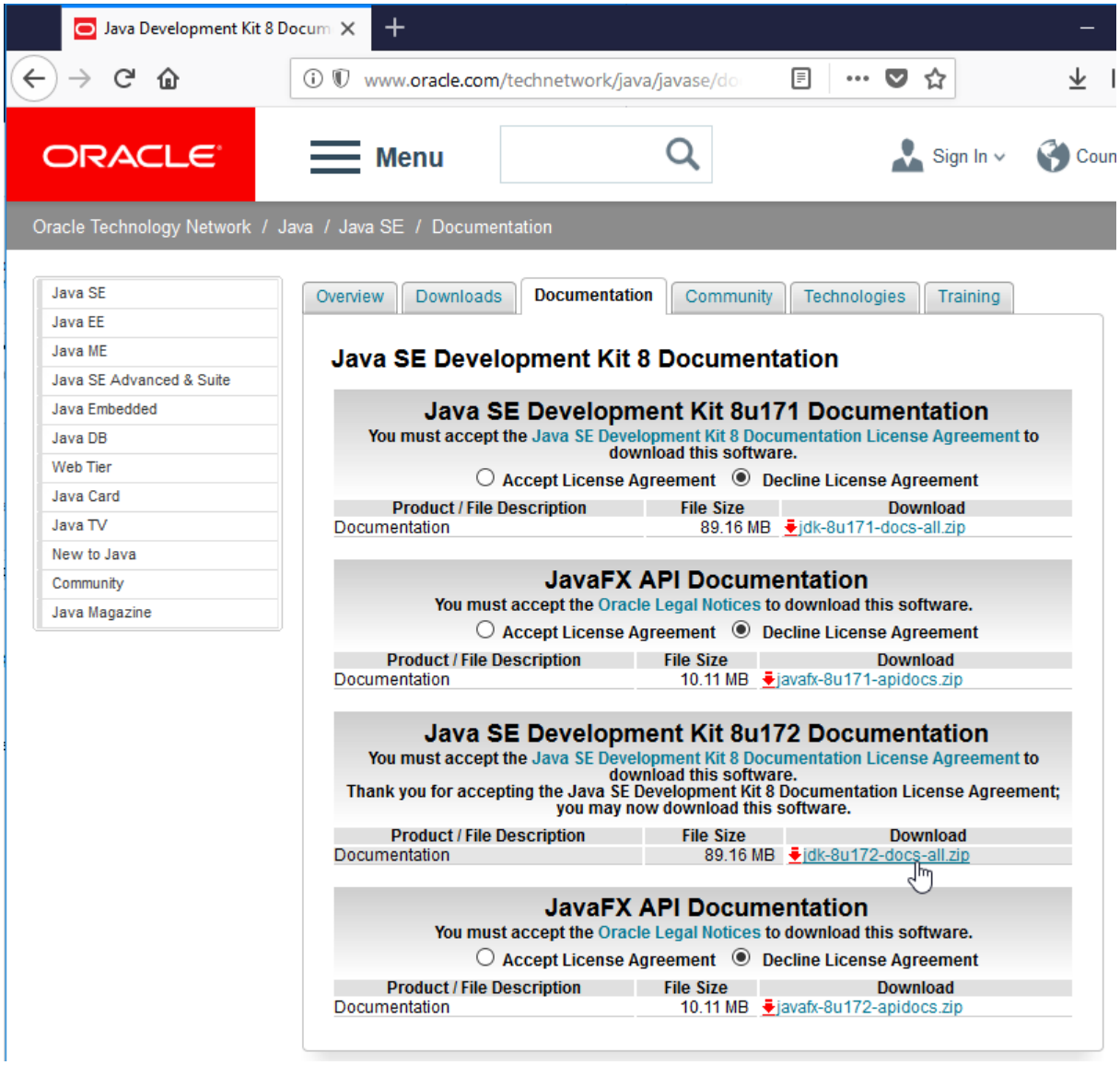


Der Download dauert etwas.

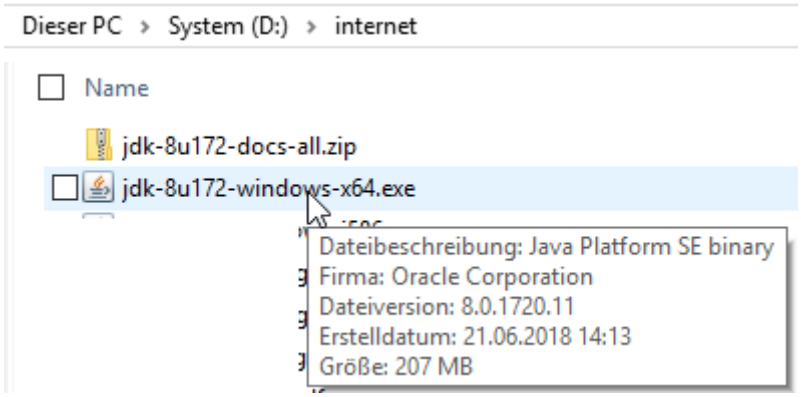
Weiterhin ist es sinnvoll, die Dokumentation zu Java, JavaDoc, weiter unten von der Auswahlseite http://www.oracle.com/technetwork/java/javase/downloads/index.html herunterzuladen, was hier nicht weiter betrachtet wird.



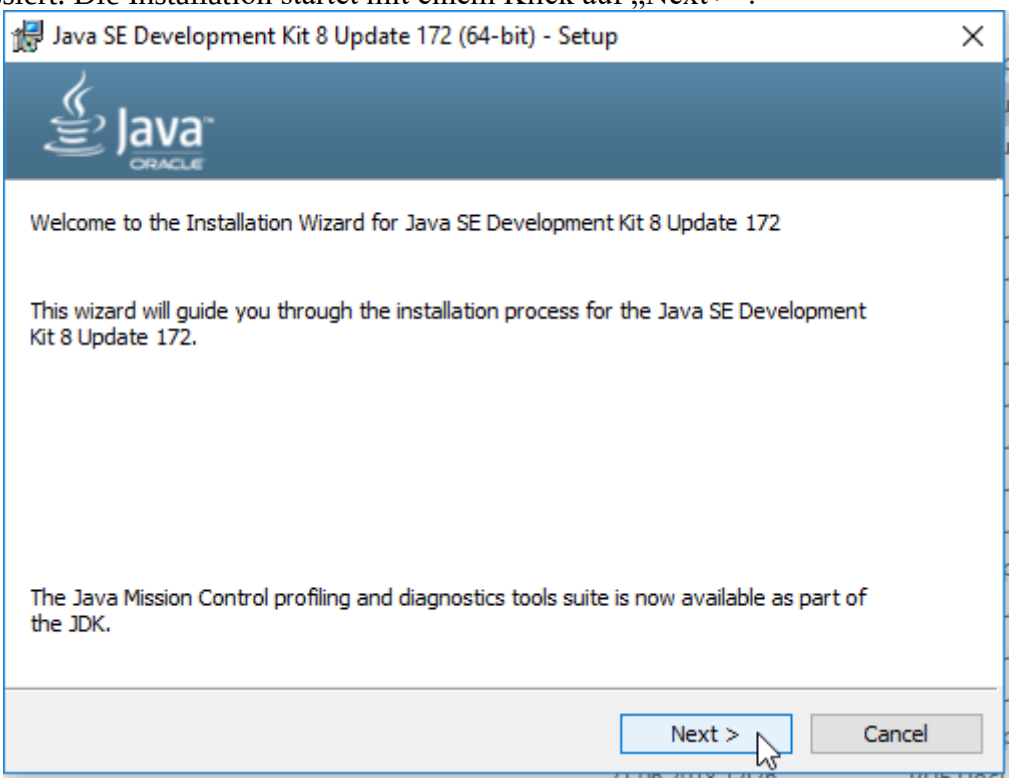
Auch hier muss zuerst das License Agreement bestätigt werden, bevotr der Download möglich ist.



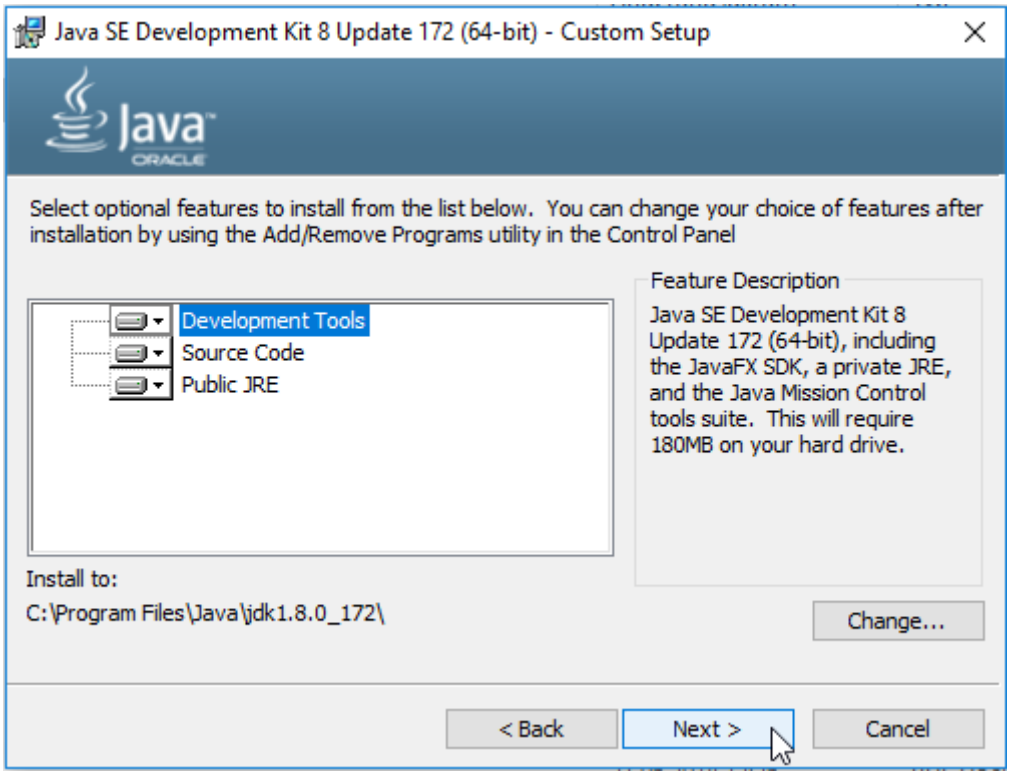
Die Installation von Java wird durch einen Doppelklick auf die heruntergeladene Datei gestartet.



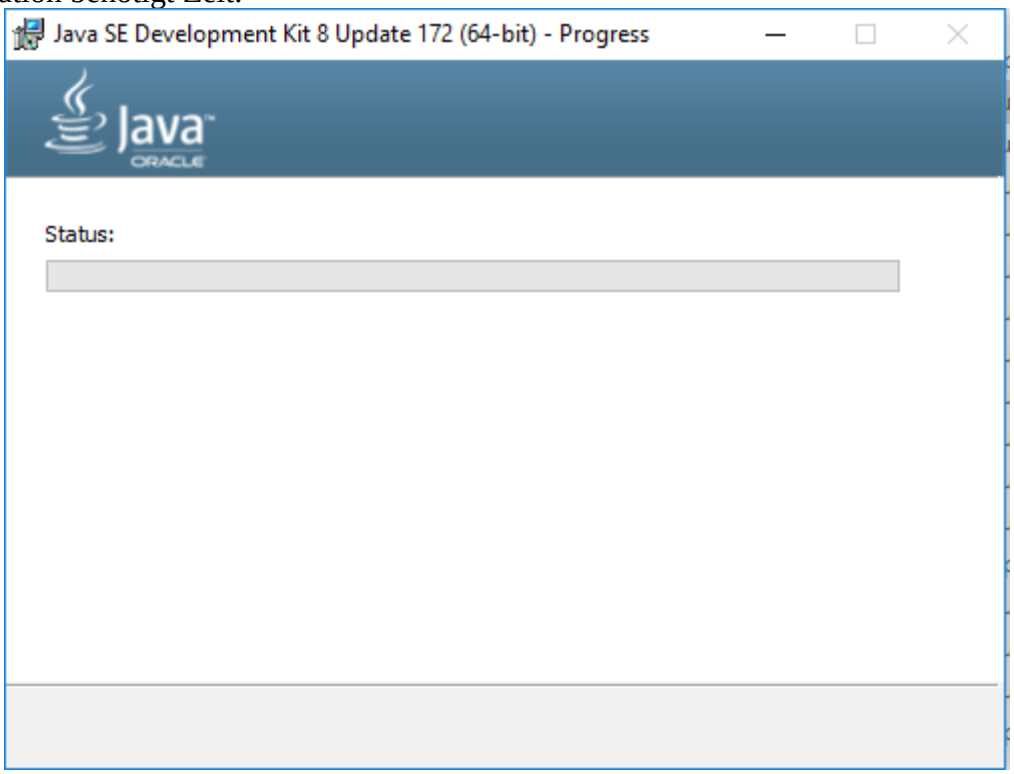
Nach einer eventuell notwendigen Bestätigung, dass die Installation durchgeführt werden soll, kann der sogenannte Installations-Wizard gestartet werden. Wichtig ist, dass in den folgenden Bildern keine Abfragen von Firewalls oder anderen individuell installierten Programmen beschrieben sind. Generell gilt, dass Java die geforderten Rechte einzuräumen sind, was natürlich trotzdem auf eigene Gefahr passiert. Die Installation startet mit einem Klick auf „Next >“.



Die Default-Einstellungen können übernommen werden, der Installationspfad (Install to) muss gemerkt werden. Es wird „Next >“ geklickt.



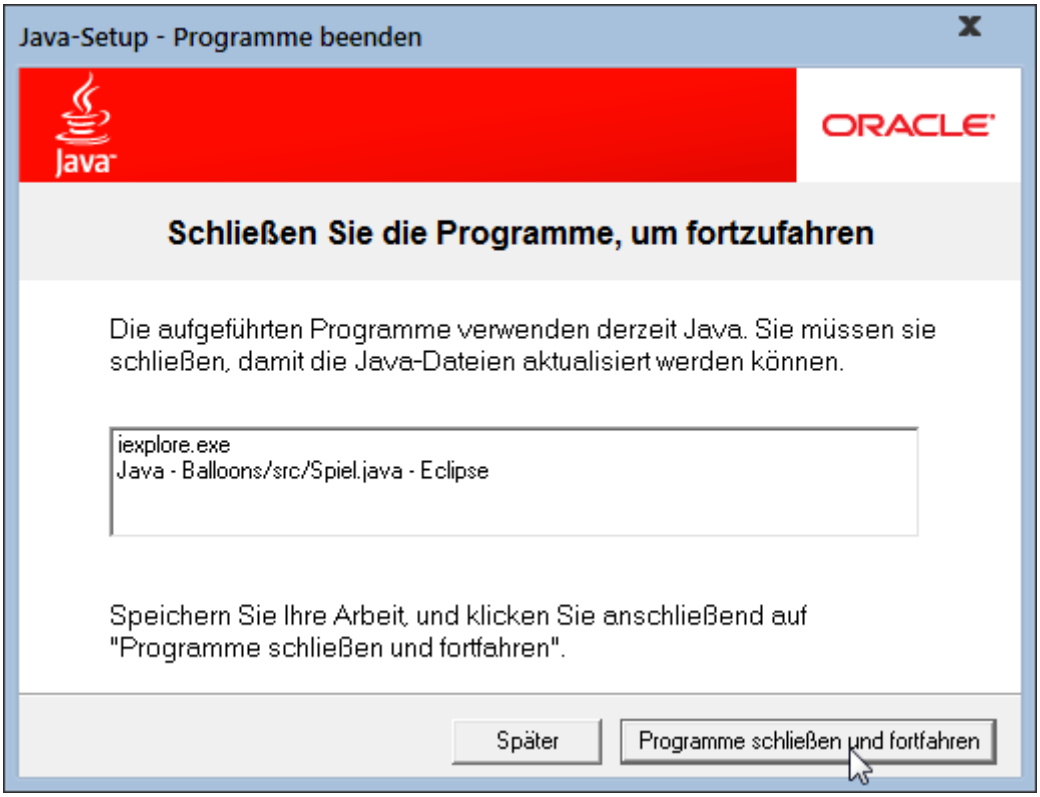
Die Installation benötigt Zeit.



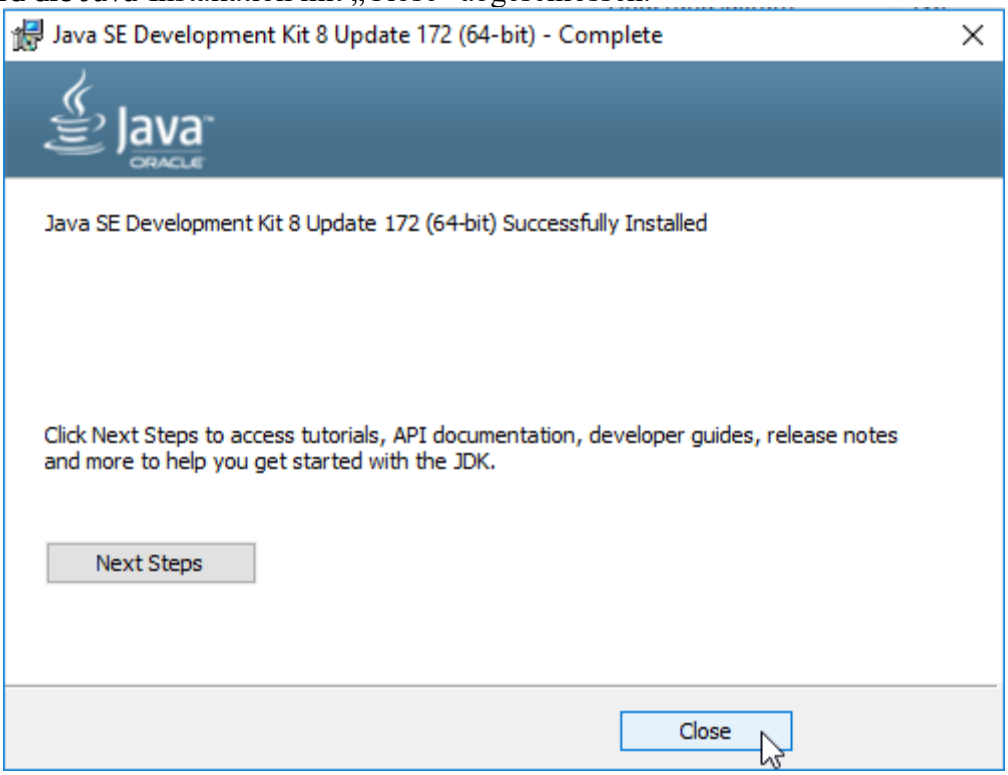
Eventuell erfolgt eine Aufforderung den Installationsort für die JRE anzugeben. Der Vorschlag wird hier mit „Weiter >“ übernommen.



Sollten noch Programme offen sein, die eine vorherige Java-Installation nutzen, sind diese zu schließen, um dann die Installation fortzusetzen.

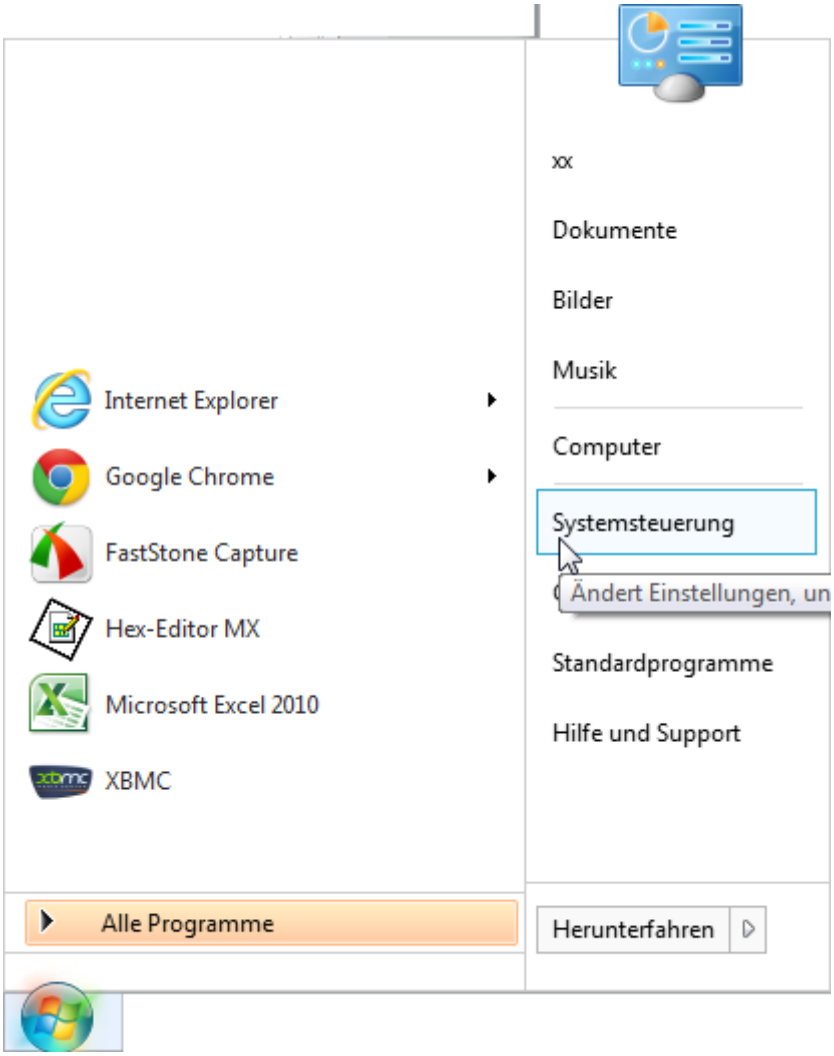


Danach wird die Java-Installation mit „Close“ abgeschlossen.

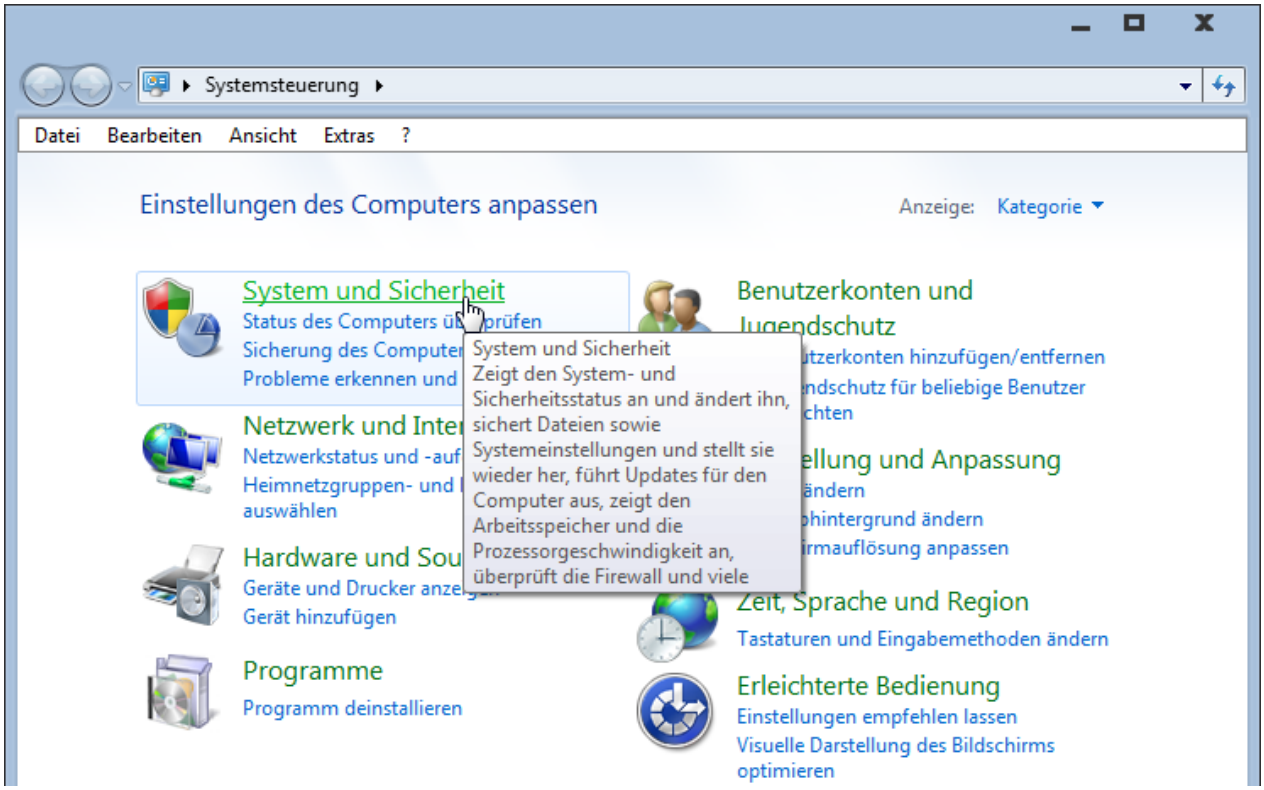


1.2 Setzen von Systemvariablen

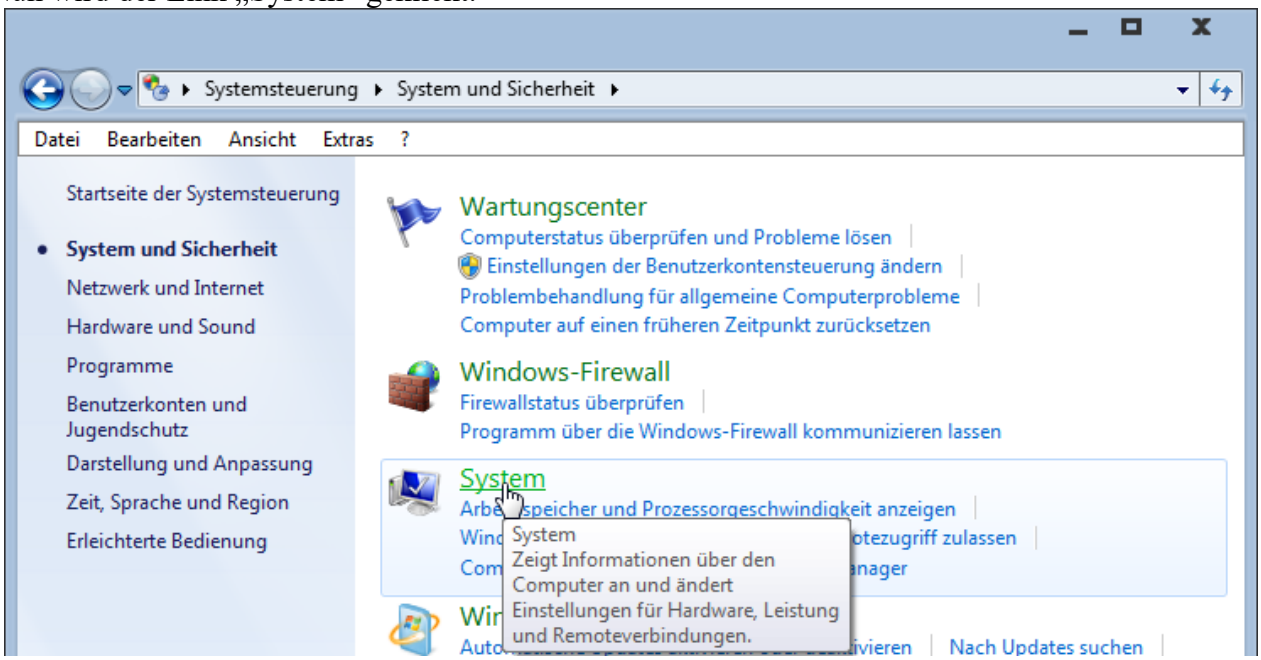
Zum Abschluss einer vollständigen Installation muss noch die Systemvariable JAVA_HOME auf das Installationsverzeichnis gesetzt werden. Ein Weg in Windows 7 dazu sieht wie folgt aus, der Weg für Windows 10 wird danach beschrieben. Zunächst wird die Systemsteuerung im Start-Menü angeklickt.



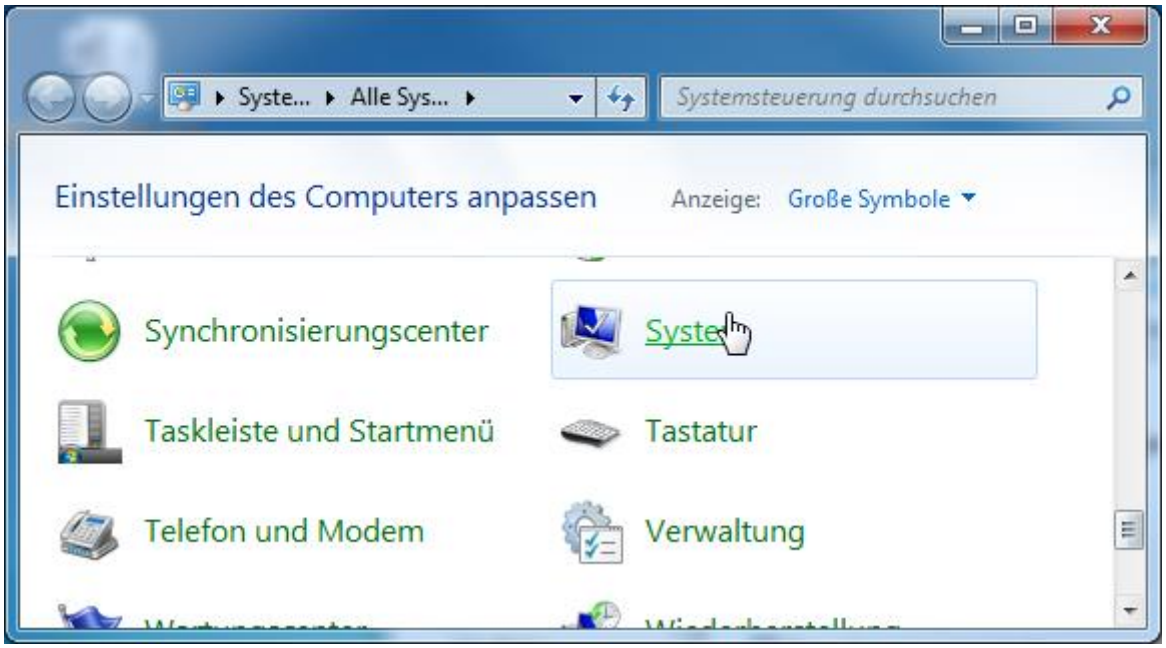
Etwas abhängig von den bisher gemachten Betriebssystemeinstellungen kann sich folgende Seite öffnen, bei der auf „System und Sicherheit“ geklickt wird. Falls die Seite anders aussehen sollte, muss bei der Möglichkeit „System“ anzuklicken, weitergelesen werden.



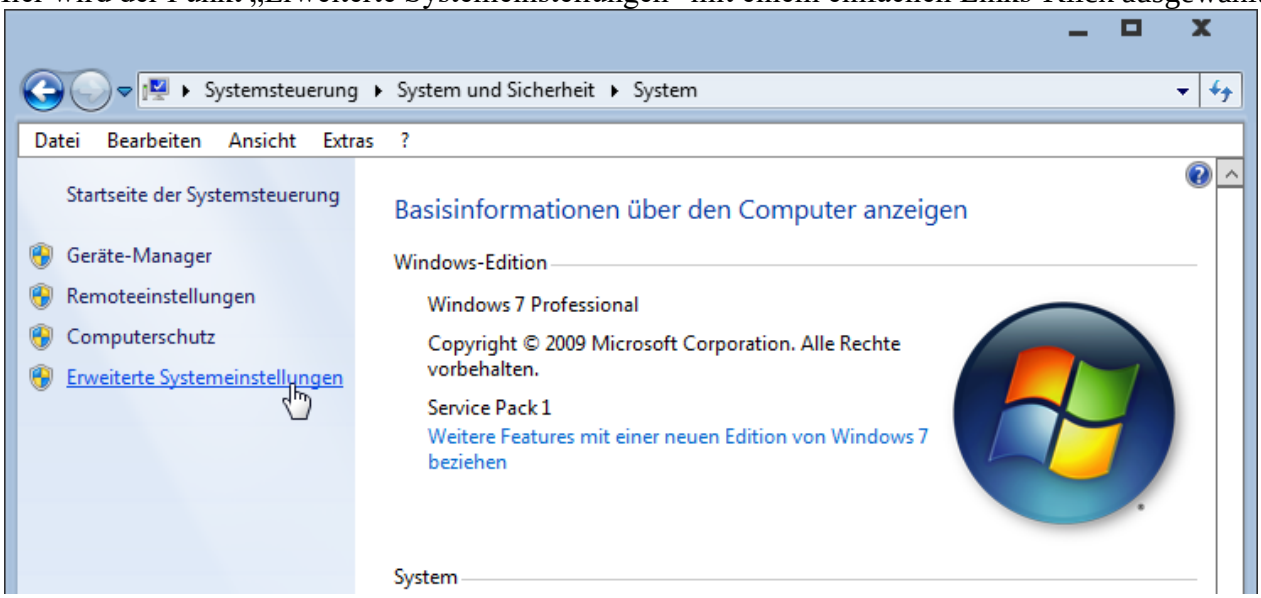
Nun wird der Link „System“ geklickt.



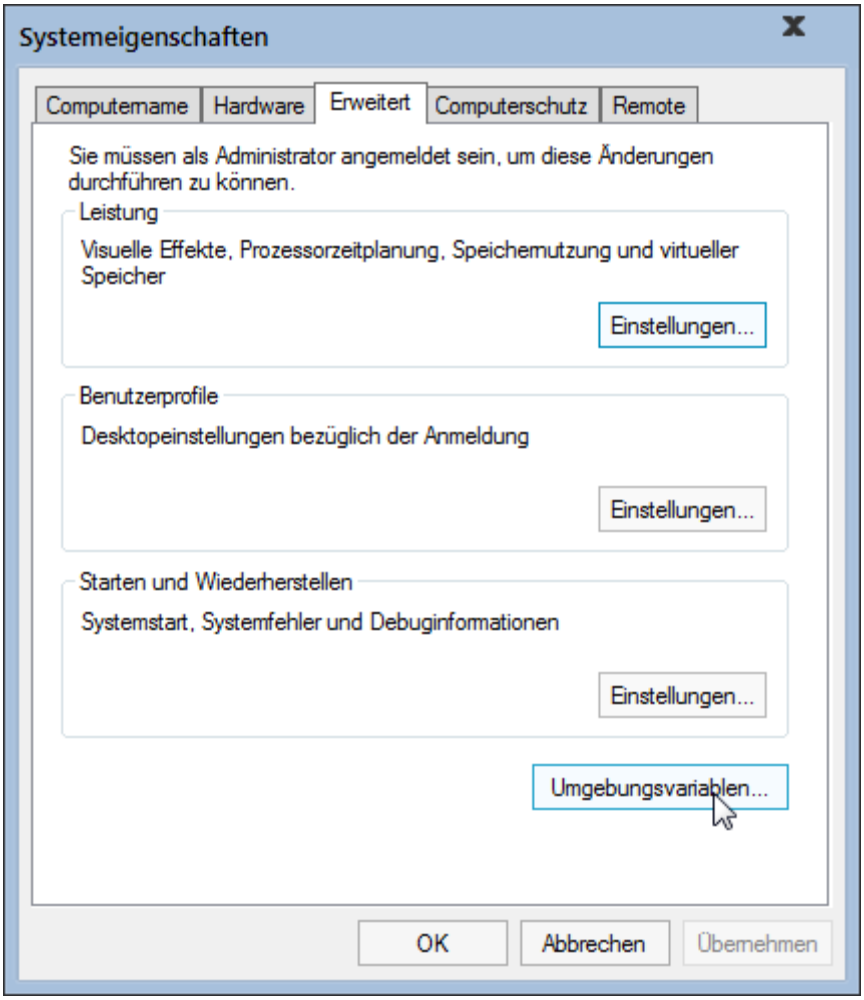
Einschub: Sollte die Seite nach der Auswahl der „Systemsteuerung“ wie folgt aussehen, wird ein Doppelklick auf „System“ durchgeführt.



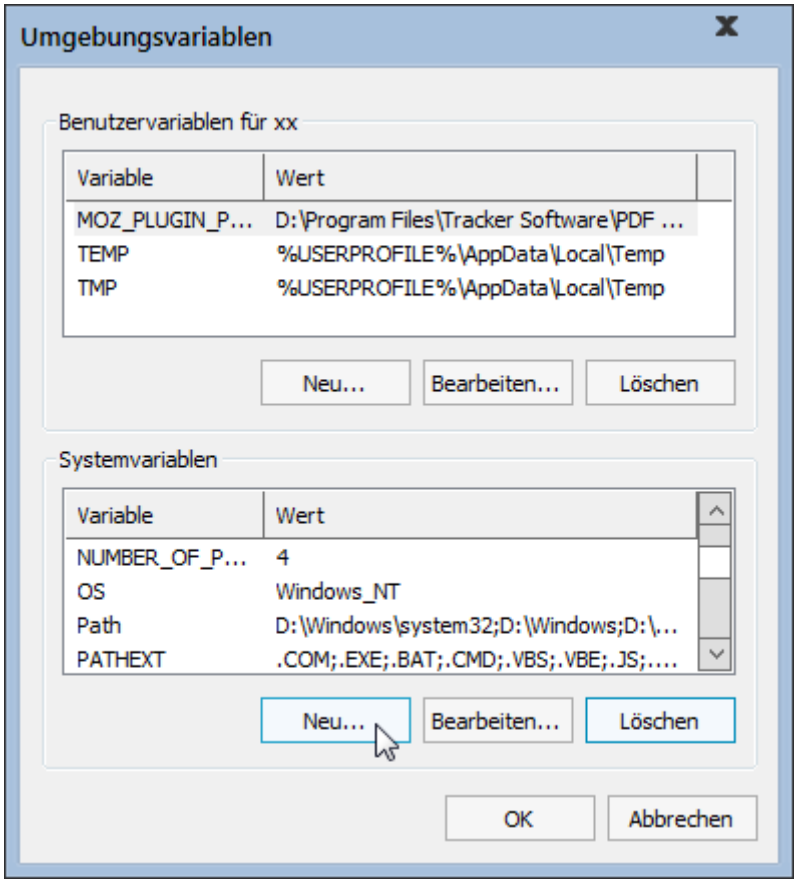
Hier wird der Punkt „Erweiterte Systemeinstellungen“ mit einem einfachen Links-Klick ausgewählt.



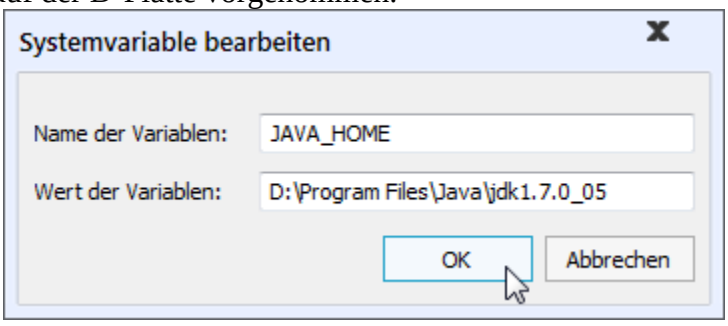
Der Reiter „Erweitert“ wird ausgewählt und unten auf „Umgebungsvariablen“ geklickt.



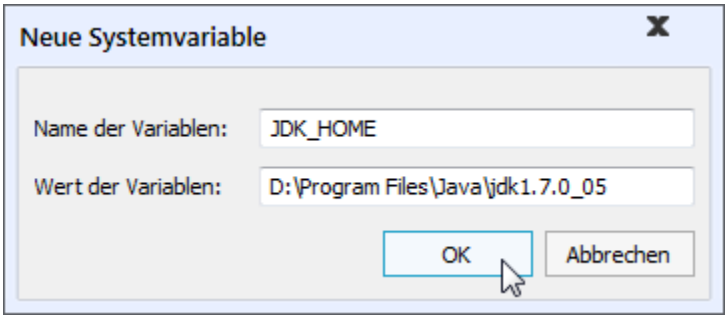
Administratoren des Systems nutzen den Knopf „Neu...“ unter Systemvariablen, andere Nutzer den Knopf „Neu…“ unter Benutzervariablen.



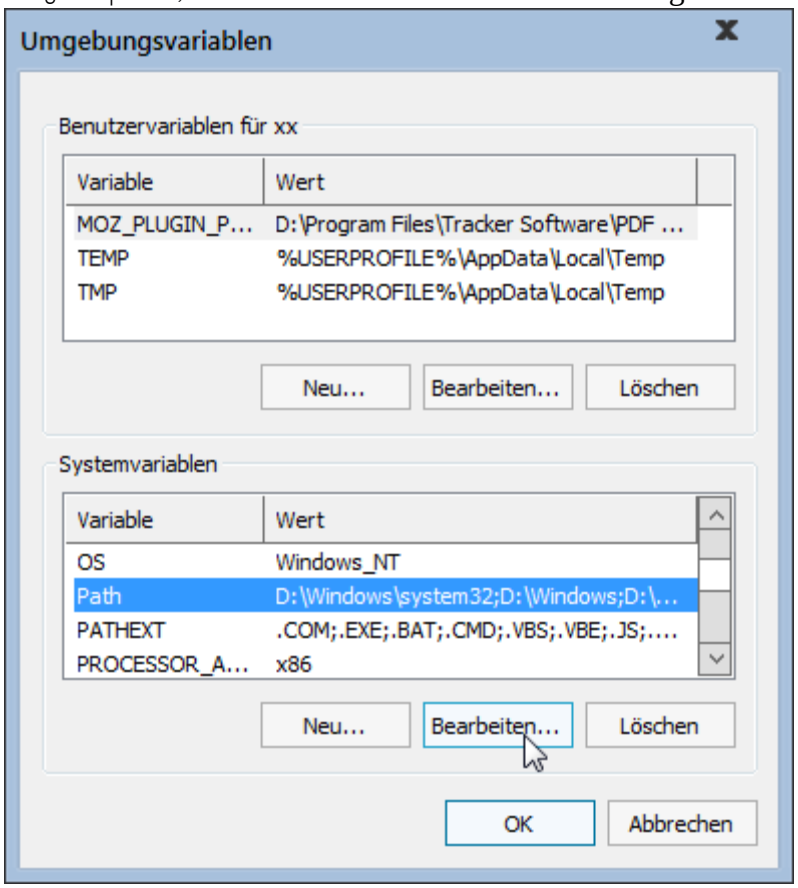
Hier werden der Variablenname sowie der Pfad zum installierten JDK eingetragen und die neue Variable mit „OK“ bestätigt. Es ist zu beachten, dass wahrscheinlich eine aktuellere Variante des JDK installiert wurde und so der letzte Ordnernamen etwas anders aussehen wird, weiterhin wurde hier eine Installation auf der D-Platte vorgenommen.



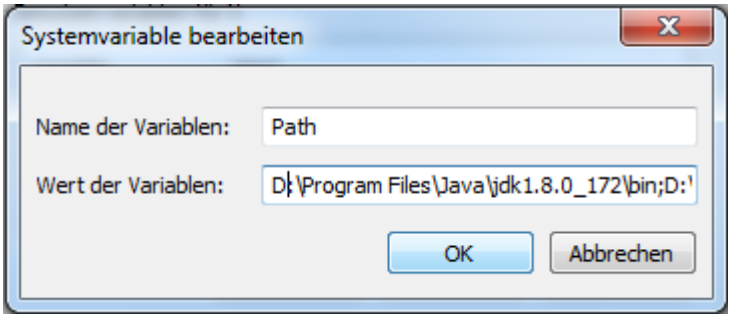
Weiterhin ist es sinnvoll, eine weitere Variable JDK_HOME mit genau dem gleichen Wert anzulegen.



Abschließend muss Java in die PATH-Variable eingetragen werden. Dazu wird die Path-Variable ausgewählt und „Bearbeiten...“ geklickt. Zunächst fällt auf, dass sich am Anfang ein Eintrag befindet, der vorher noch nicht da war. Dieser Eintrag `C:\Program Files\Common Files\Oracle\Java\javapath;` wird zusammen mit dem Semikolon gelöscht.



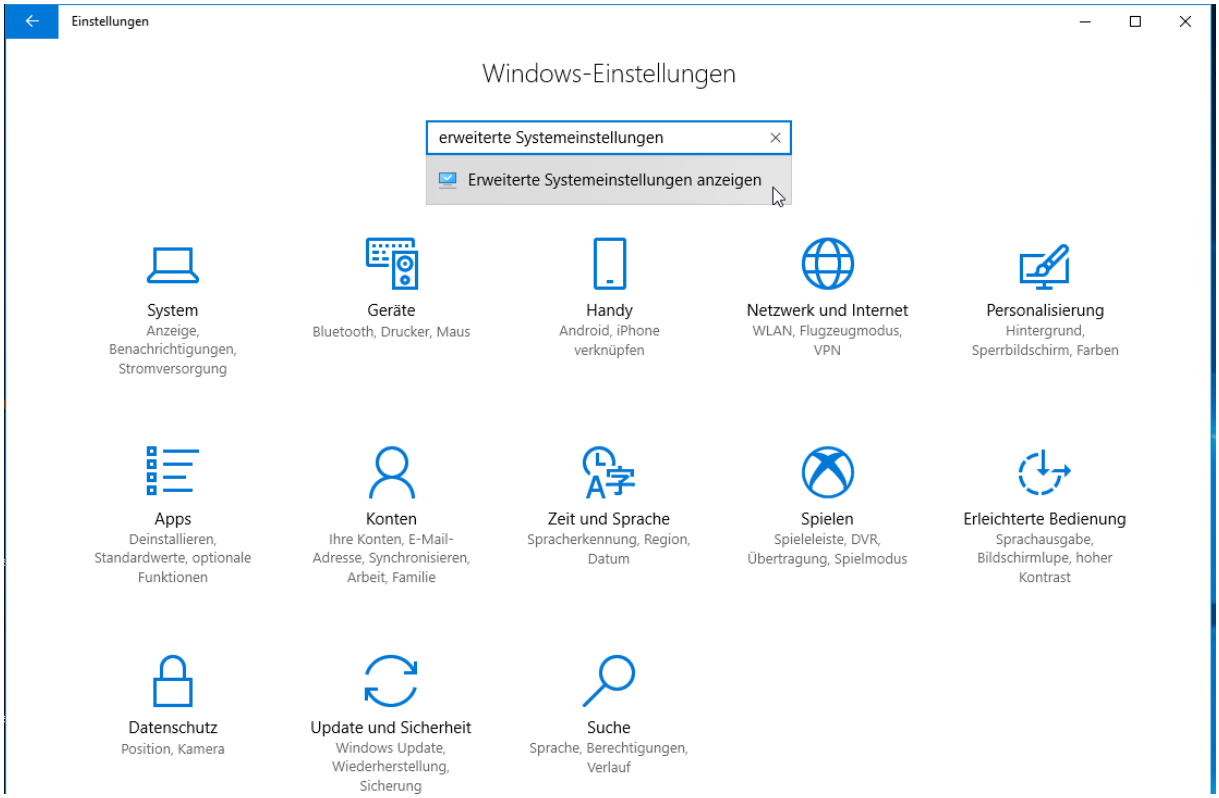
Danach muss der Pfad irgendwo, z. B. am Anfang, um das Java-Verzeichnis verlängert um \bin (`D:\Program Files\Java\jdk1.8.0_172\bin`) ergänzt werden. Es ist zu beachten, dass alle Einträge in Path mit einem Semikolon getrennt sind, wobei am Ende kein Semikolonstehen darf. Im Beispiel wurde der Pfad endend mit einem Semikolon vorne angehängt. Danach werden alle Änderungen mehrfach über „OK“ bestätigt.



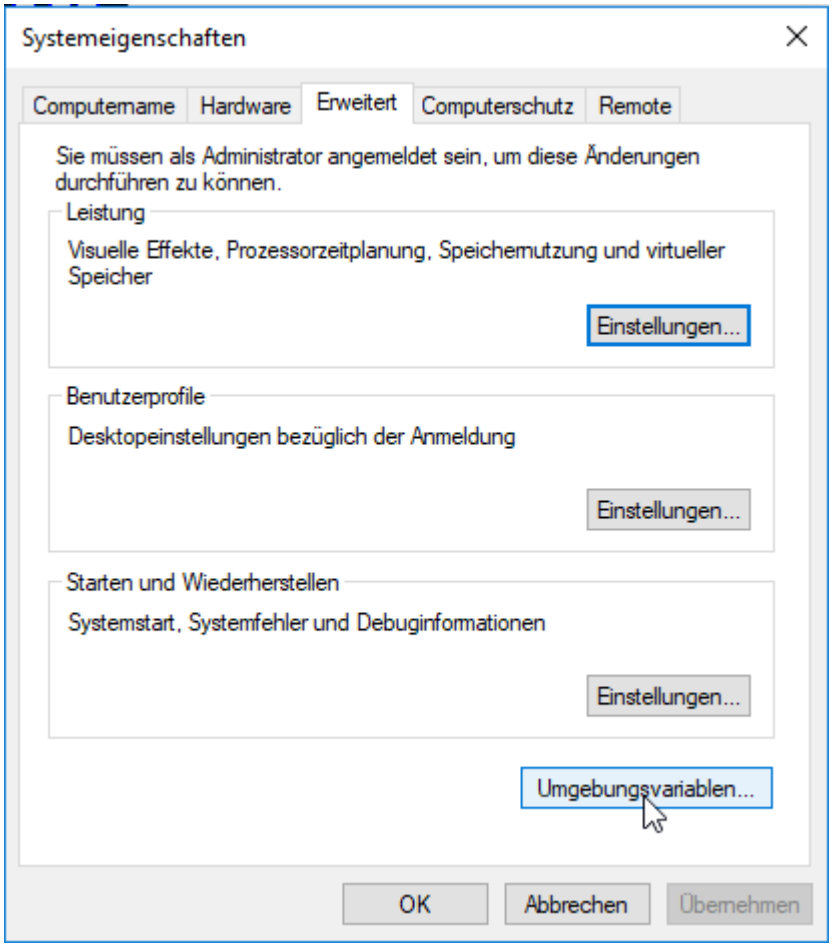
Bei Windows 10 beginnt ein Weg mit einem Klick auf dem Start-Icon und der Auswahl „Einstellungen“.



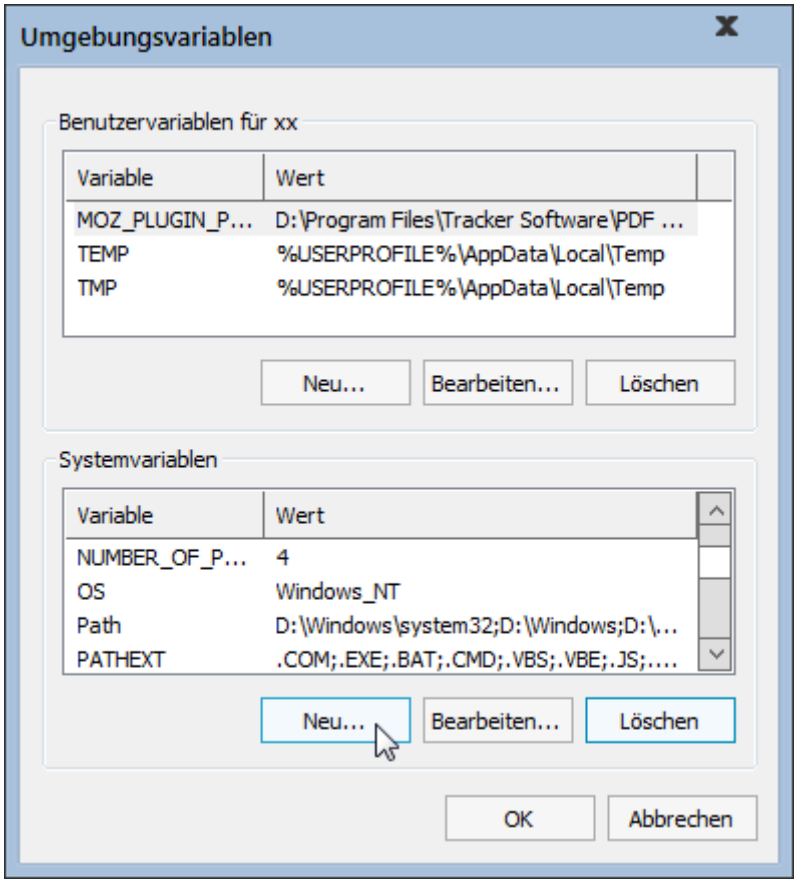
Im oberen Suchfenster wird „erweiterte Systemeinstellungen eingeben und darunter dann mit einem Links-Klick ausgewählt.



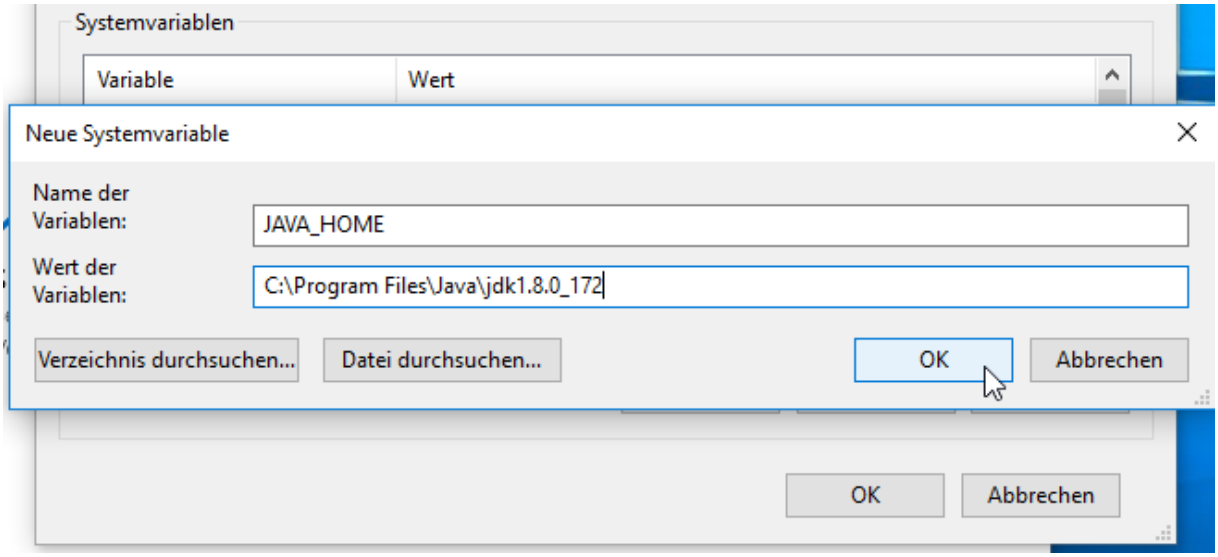
Es wird auf „Umgebungsvariablen“ geklickt.



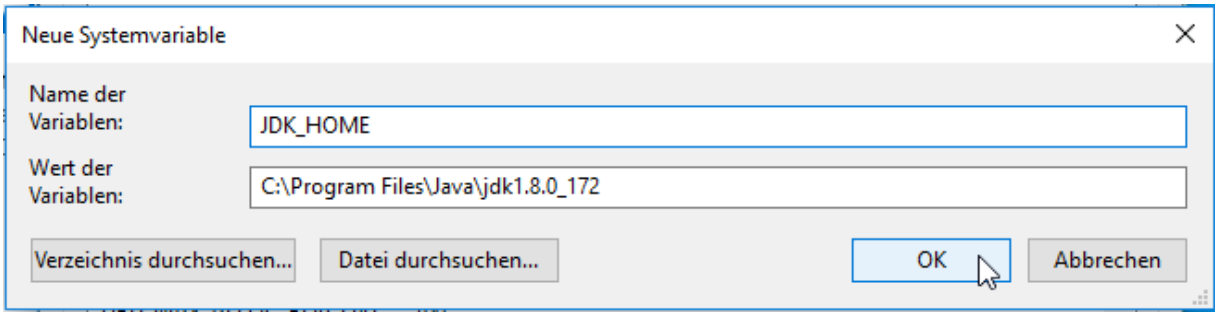
Administratoren des Systems nutzen den Knopf „Neu...“ unter Systemvariablen, andere Nutzer den Knopf „Neu…“ unter Benutzervariablen.



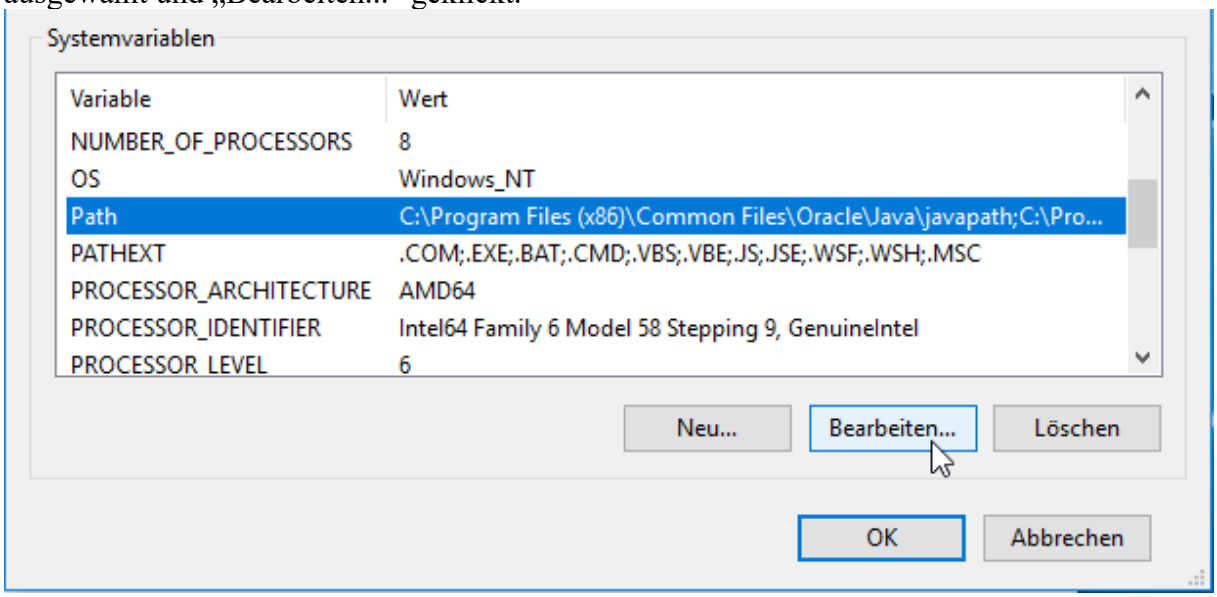
Hier werden der Variablenname sowie der Pfad zum installierten JDK eingetragen und die neue Variable mit „OK“ bestätigt. Es ist zu beachten, dass wahrscheinlich eine aktuellere Variante des JDK installiert wurde und so der letzte Ordnernamen etwas anders aussehen wird.



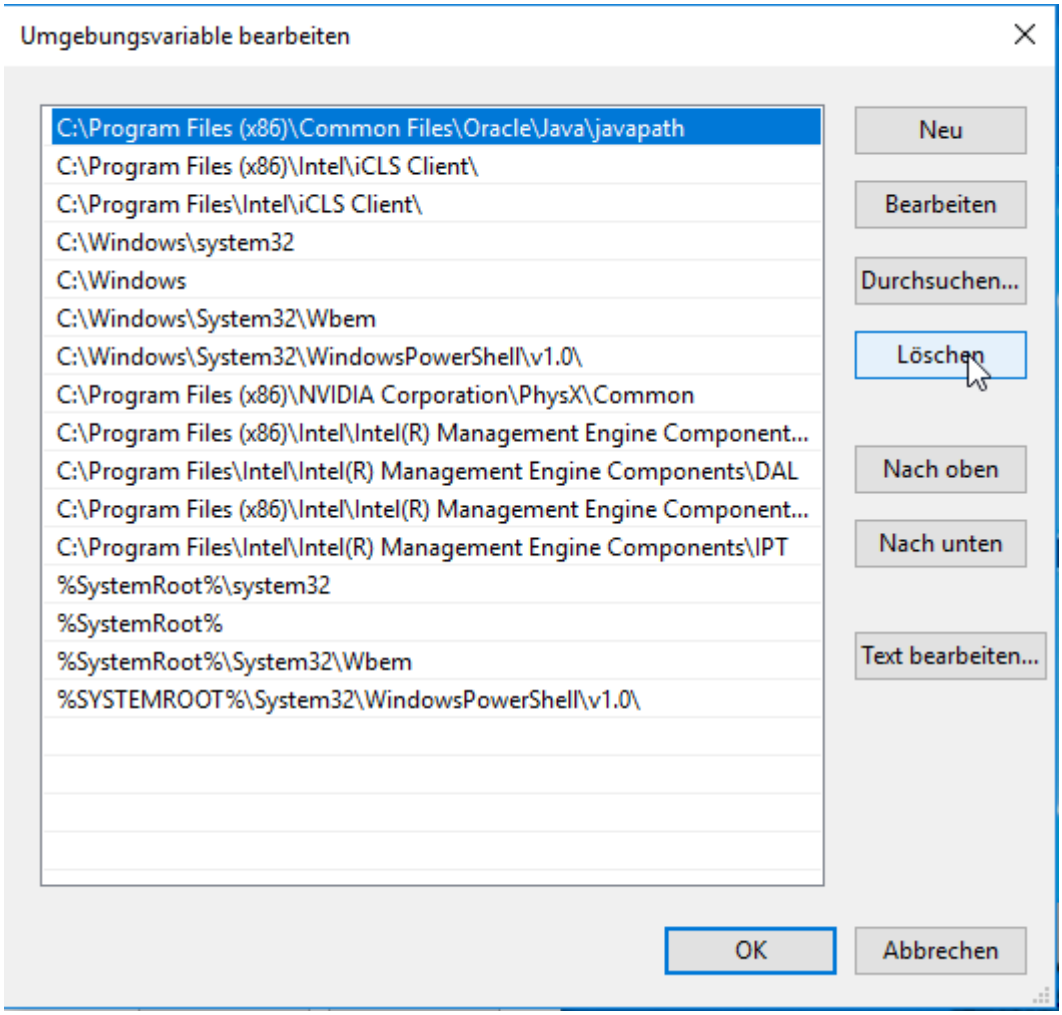
Weiterhin ist es sinnvoll, eine weitere Variable JDK_HOME mit genau dem gleichen Wert anzulegen.



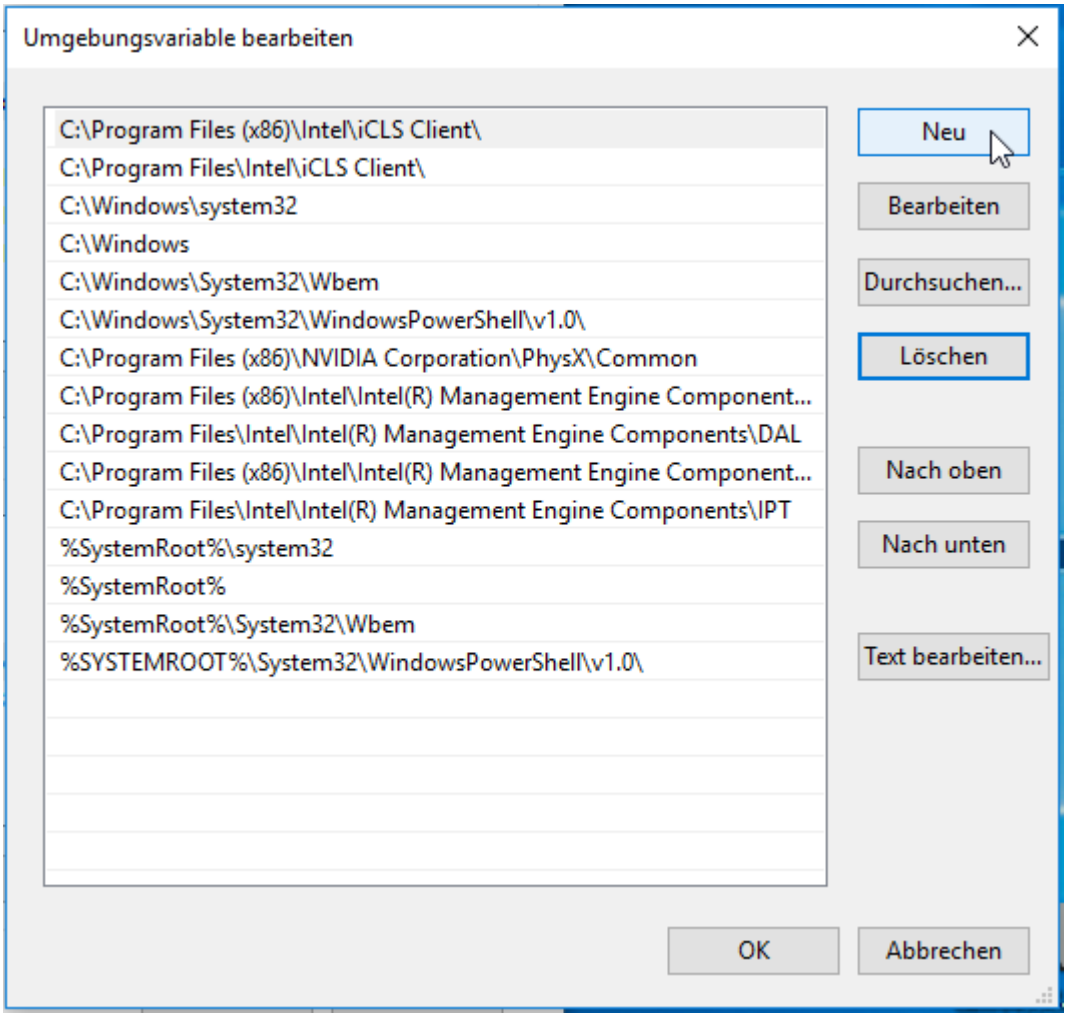
Abschließend muss Java in die PATH-Variable eingetragen werden. Dazu wird die Path-Variable ausgewählt und „Bearbeiten...“ geklickt.



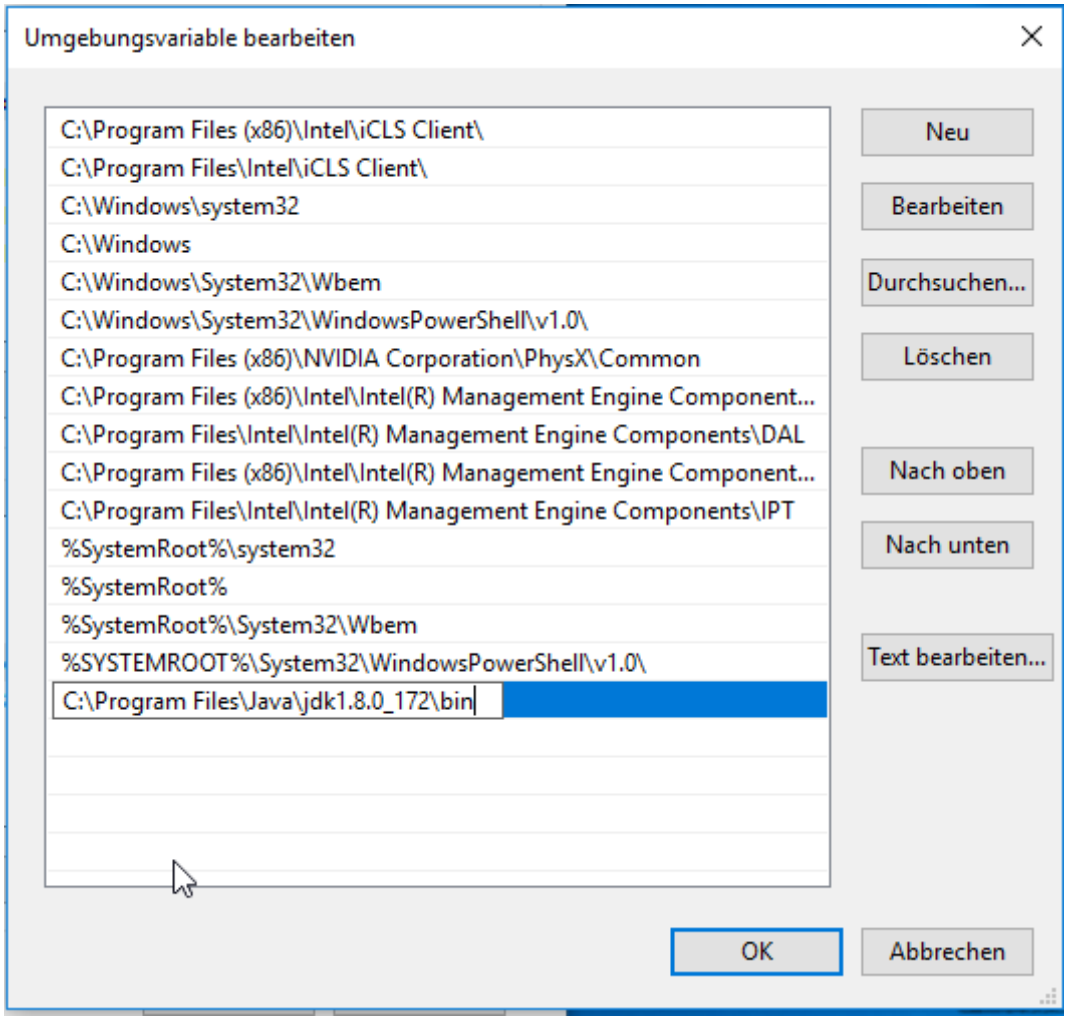
Zunächst fällt auf, dass sich am Anfang ein Eintrag befindet, der vorher noch nicht da war. Dieser Eintrag `C:\Program Files\Common Files\Oracle\Java\javapath` (evtl. mit der Ergänzung „ (x86)“) wird mit einem Links-Klick auf „Löschen“ gelöscht.



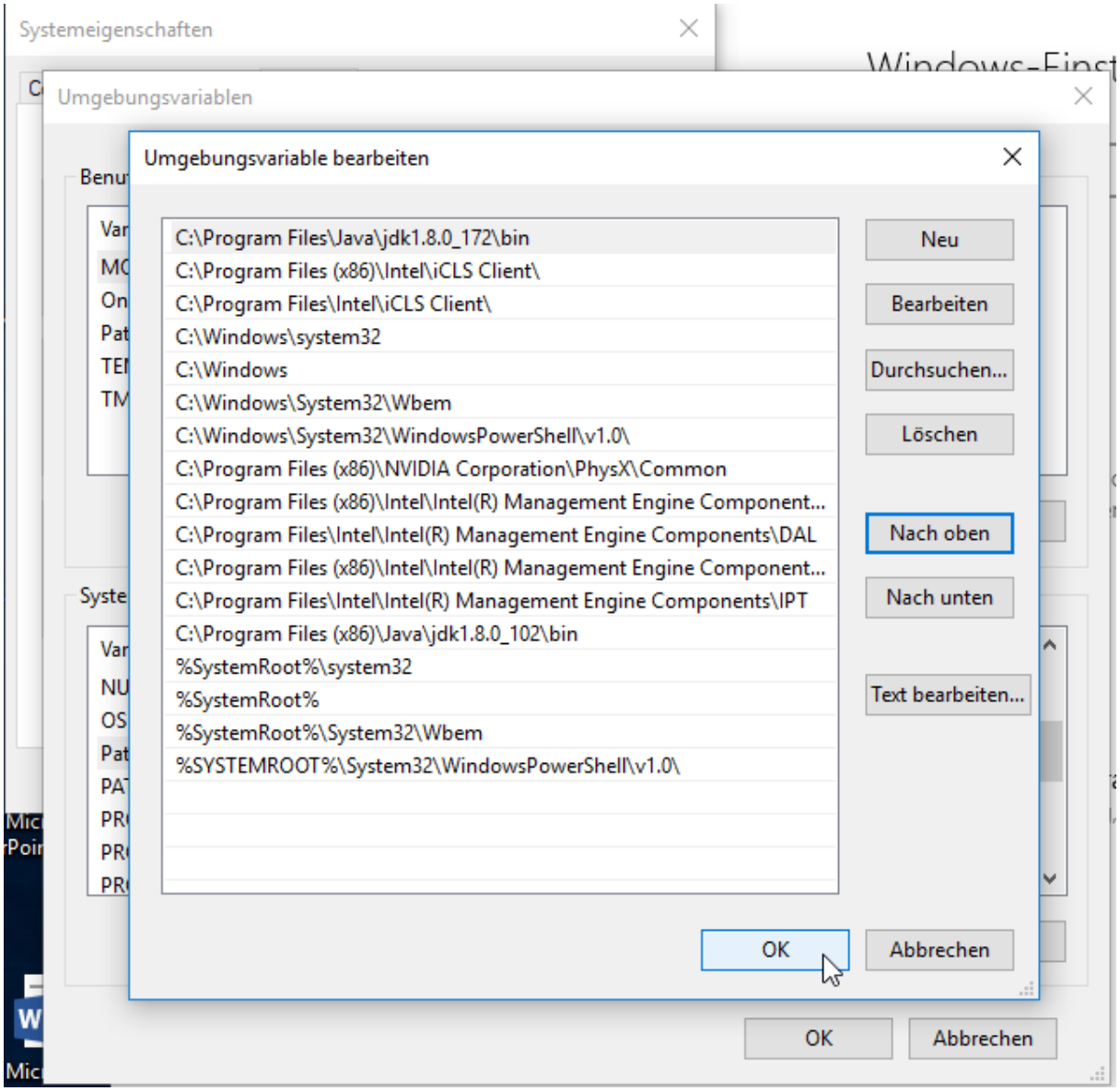
Danach wird auf „Neu“ oben geklickt.



Es wird der Pfad zum bin-Verzeichnis `C:\Program Files\Java\jdk1.8.0_172\bin` eingetragen.



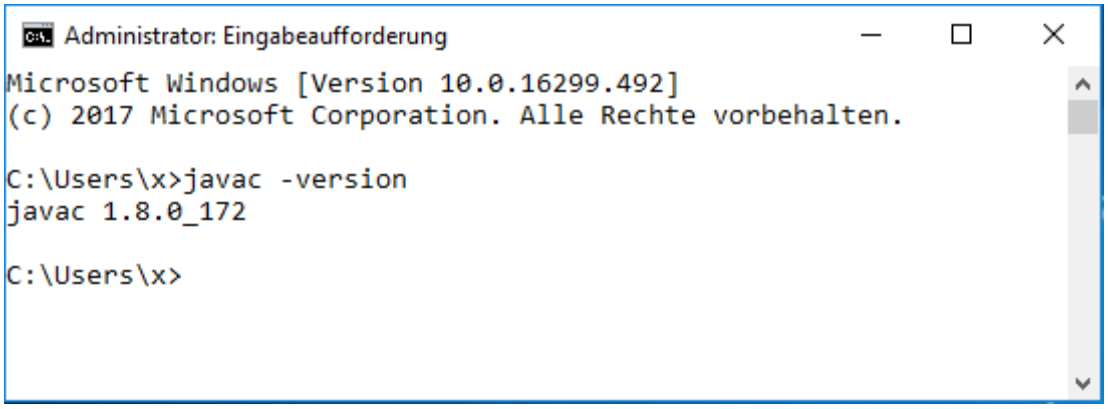
Danach kann der ausgewählte Pfad durch mehrfaches Klicken von „Nach oben“ in die obere Zeile geschoben werden, was allerdings für die Nutzung nicht relevant ist. Die Eintragung wird jeweils mit „OK“ bestätigt.



Zur Überprüfung der Installation kann ein Konsolenfenster genutzt werden. Dies ist z. B. über einen Klick auf sas Windows-Icon links-unten, dem Scrollen zum Buchstaben „W“ bei den installierten Programmen, dem Aufklappen von „Windows-System“ und der Auswahl von „Eingabeaufforderung“ erreichbar

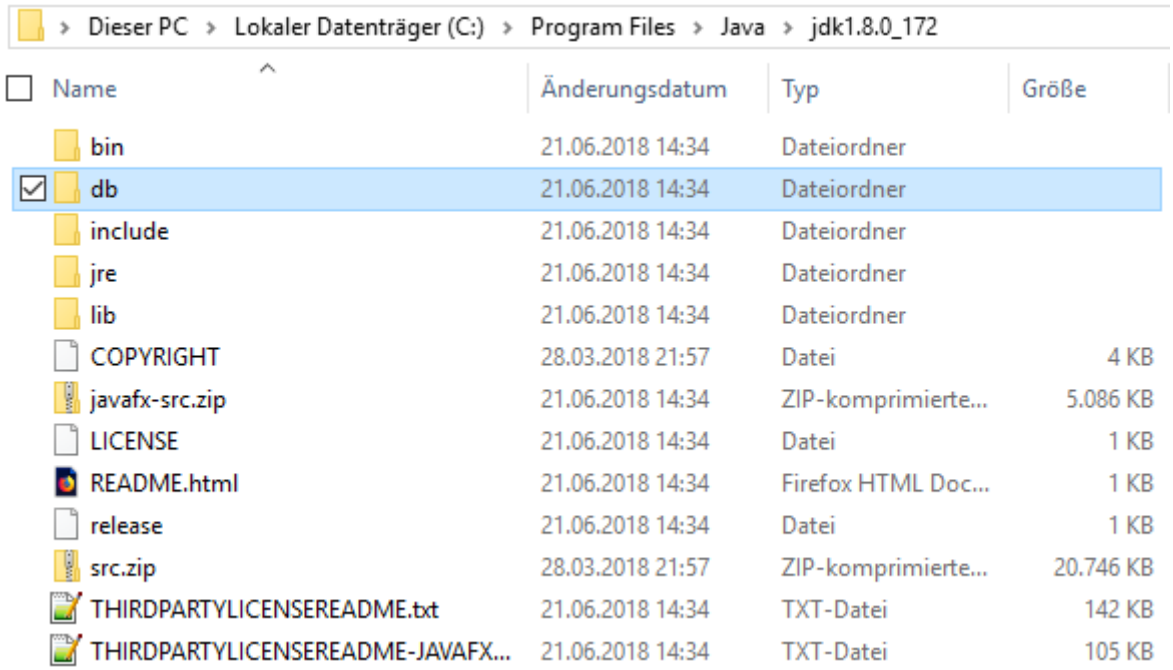


In dem Fenster wird `javac -version` eingegeben, was zur angezeigten Ausgabe führen muss. Java ist dann korrekt installiert.



1.3 Installation von Apache Derby (früher Java DB)

Achtung, im JDK 8u181 verschwand die sonst immer mit installierte Java DB im Juli 2018 ohne Begründung aus dem Installationsverzeichnis (http://www.oracle.com/technetwork/java/javase/8u181-relnotes-4479407.html). Bis dahin existierte ein Verzeichnis db im JDK-Installationsverzeichnis.



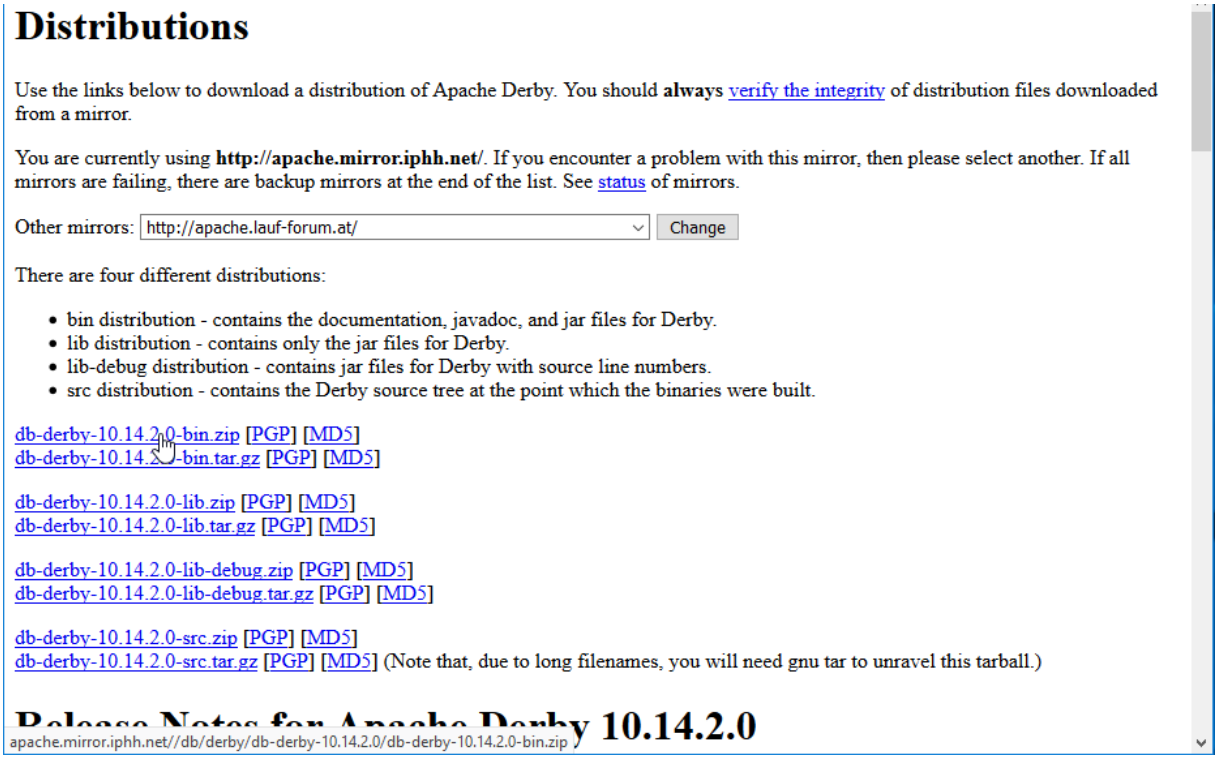
Existiert dieses Verzeichnis, muss keine weitere Installation stattfinden, sonst muss Apache Derby zusätzlich installiert werden. Die Installation beginnt mit einem Klick auf den Download-Reiter auf der Webseite https://db.apache.org/derby/.



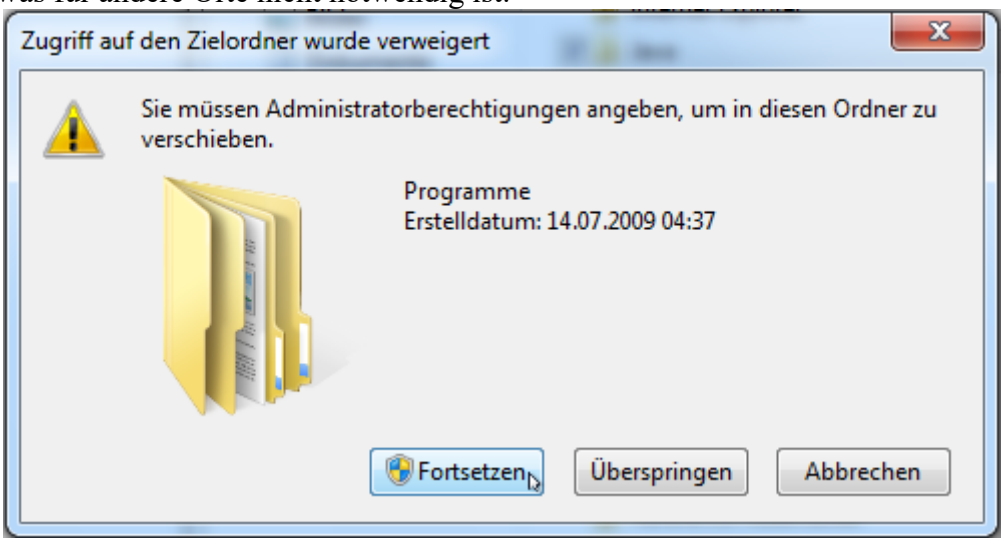
Es wird eine aktuelle zur installierten Java-Version passende Derby-Version angeklickt.



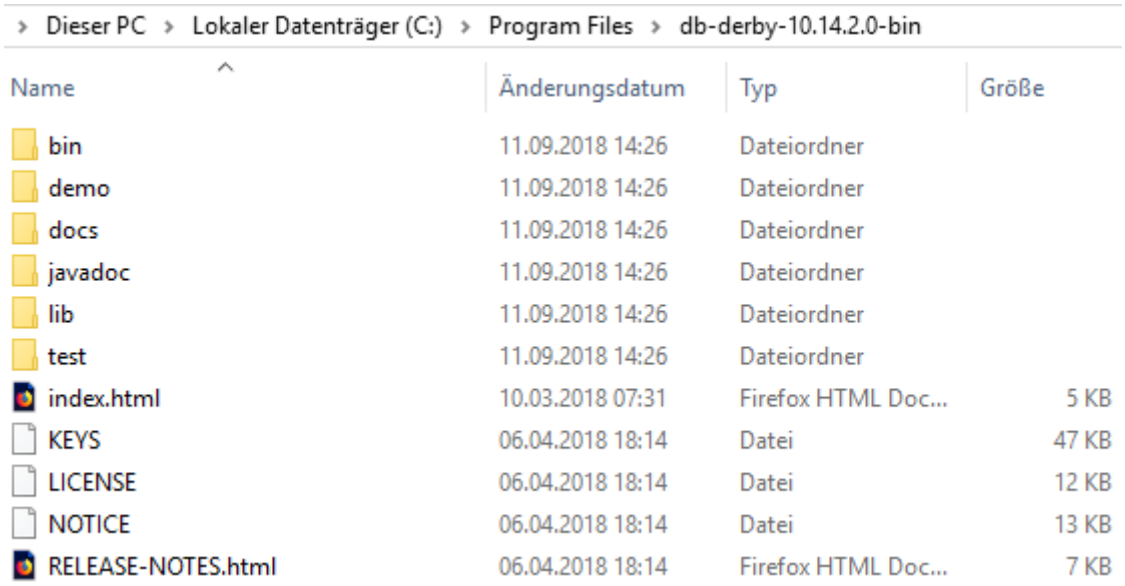
Es wird die aktuelle …bin.zip heruntergeladen.



Die Zip-Datei wird ausgepackt und das entstehende Verzeichnis an einen sinnvollen Ort geschoben, hier z. B. C:\Programme (oder C:\Program Files). Für diesen Zielort müssen Administatorrechte vorliegen, was für andere Orte nicht notwendig ist.



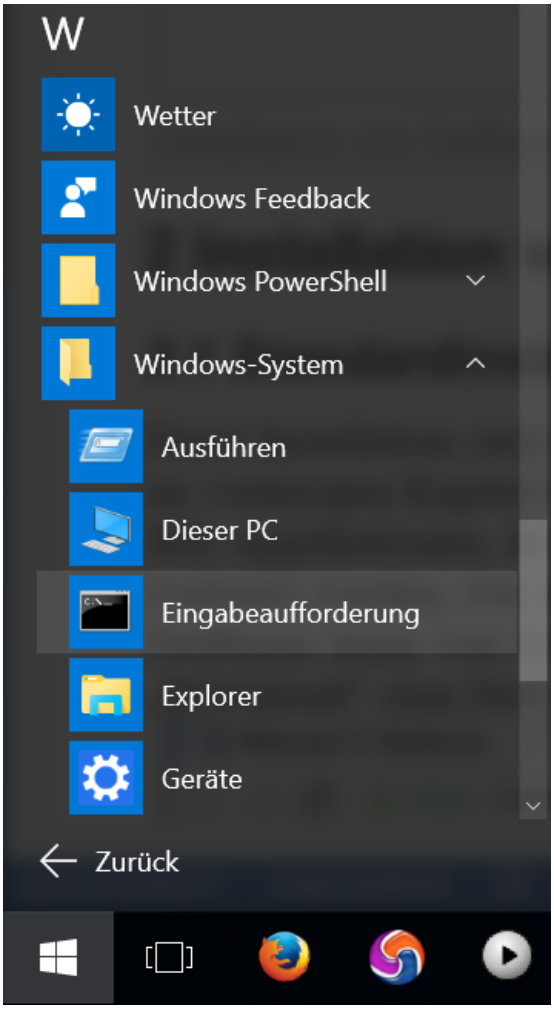
Das entstehende Verzeichnis sieht wie folgt aus.



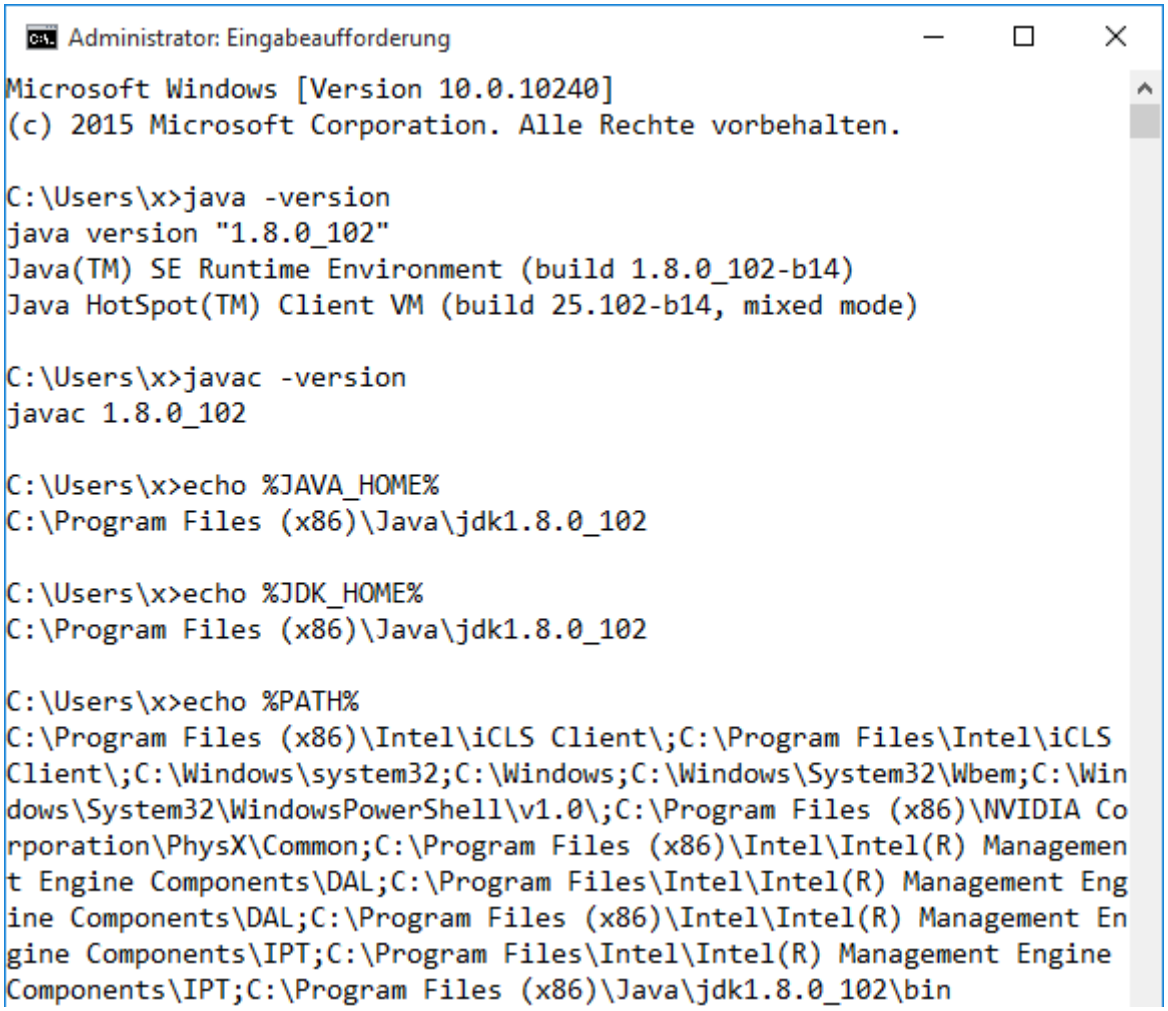
Soll intensiver direkt mit Derby gearbeitet werden, ist die Aufnahme des bin-Verzeichnisses in die PATH-Variable von Windows sinnvoll, wird aber sonst nicht benötigt. Bei direkter Nutzung muss die Systemvariable DERBY_HOME auf das Installationsverzeichnis gesetzt werden.

1.4 Überprüfung der Installation

Zur Validierung, welche Java-Version installiert ist, kann ein Konsolen-Fenster genutzt werden, das unter Windows 10 unter „Alle Apps“, „Windows-System“ und „Eingabeaufforderung“ erreichbar ist. Die Windows-PowerShell ist genauso nutzbar.



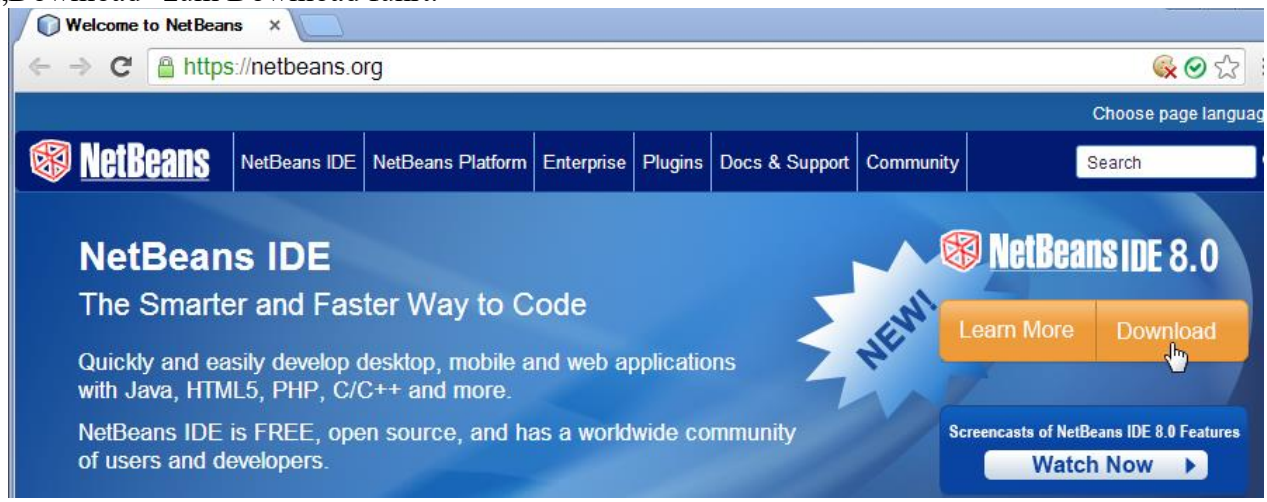
Die Ausgaben zu den Aufrufen `java -version` und `javac -version` geben die installierten Java-Versionen an. Sollte es eine Fehlermeldung geben, ist Java nicht korrekt installiert, was z. B. an einer Fehlerhaften PATH-Variablen liegen kann. Die PATH-Variable ist z. B. mit `echo %PATH%` ausgebbar.


Administrator: Eingabeaufforderung
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. Alle Rechte vorbehalten.
C:\Users\x>java -version
java version "1.8.0_102"
Java(TM) SE Runtime Environment (build 1.8.0_102-b14)
Java HotSpot(TM) Client VM (build 25.102-b14, mixed mode)
C:\Users\x>javac -version
javac 1.8.0_102
C:\Users\x>echo %JAVA_HOME%
C:\Program Files (x86)\Java\jdk1.8.0_102
C:\Users\x>echo %JDK_HOME%
C:\Program Files (x86)\Java\jdk1.8.0_102
C:\Users\x>echo %PATH%
C:\Program Files (x86)\Intel\iCLS Client\;C:\Program Files\Intel\iCLS
Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Win
dows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\NVIDIA Co
rporation\PhysX\Common;C:\Program Files (x86)\Intel\Intel(R) Managemen
t Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Eng
ine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management En
gine Components\IPT;C:\Program Files\Intel\Intel(R) Management Engine
Components\IPT;C:\Program Files (x86)\Java\jdk1.8.0_102\bin

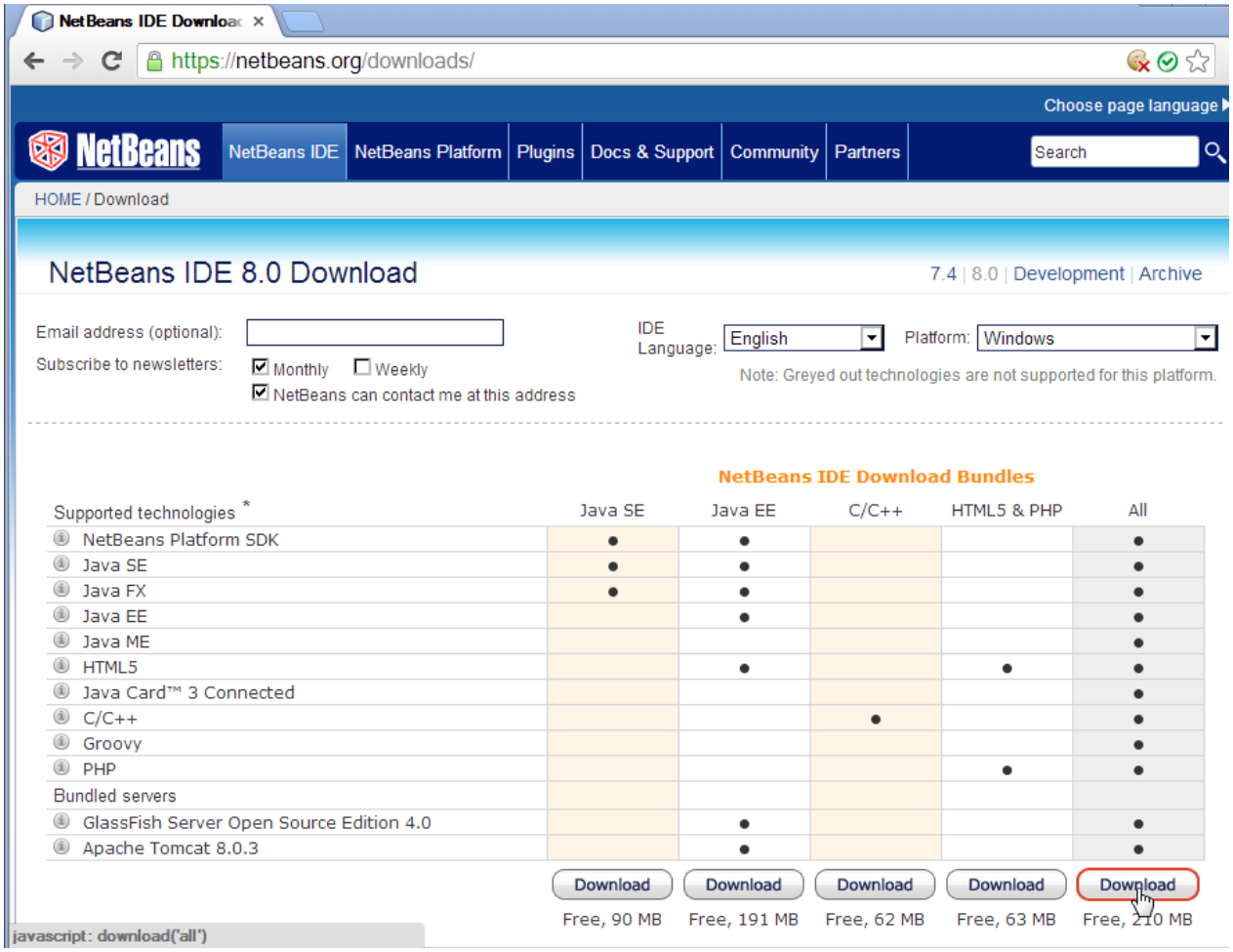
2 Installation von NetBeans (NetBeans 8)

2.1 Standardinstallation

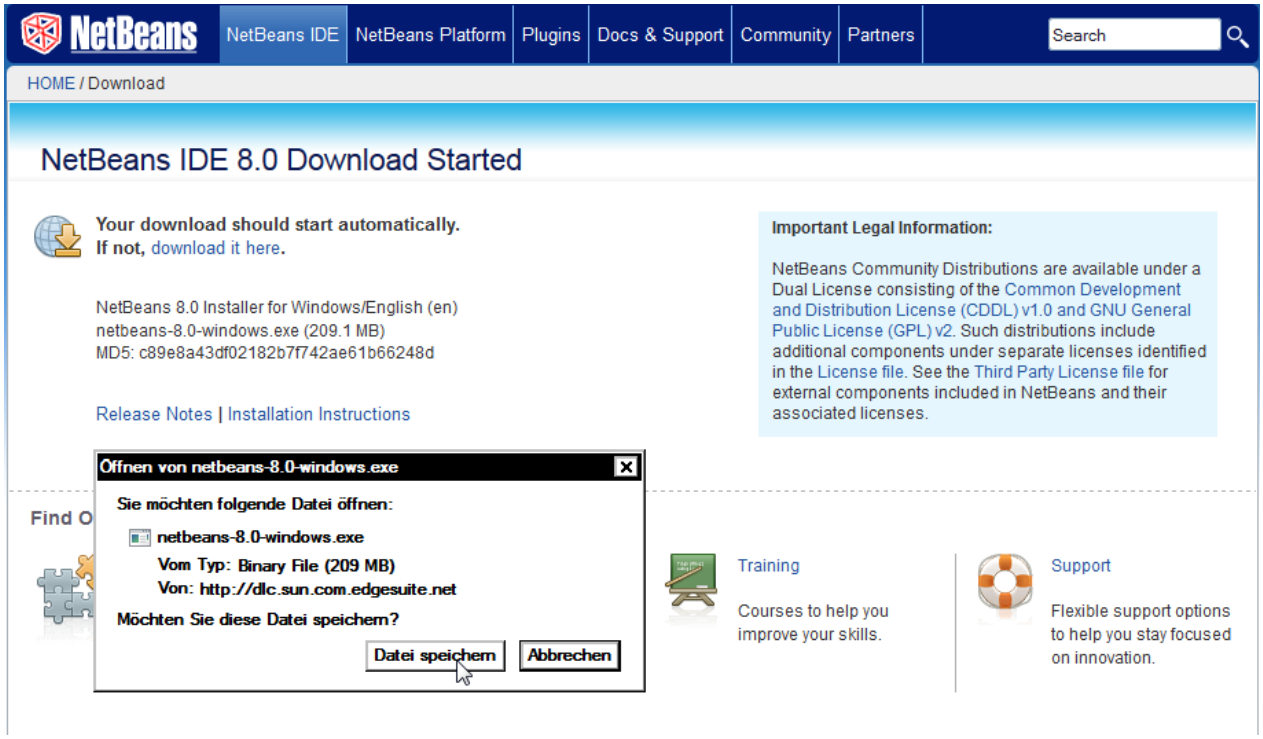
Diese Installation setzt voraus, dass Java ab der Version 8 (update 60) zusammen mit Java DB, wie im vorherigen Kapitel beschrieben, installiert ist. Ziel der Installation ist die einfache Erstellung von JEE-Applikationen, die auf einem lokalen Server (GlassFish) laufen. Für reale Projekte muss getrennt evaluiert werden, wie die Software auf dem jeweiligen Server installiert werden muss.
NetBeans kann von folgender Seite http://netbeans.org/ bezogen werden, auf der z. B. der Knopf „Download“ zum Download führt.



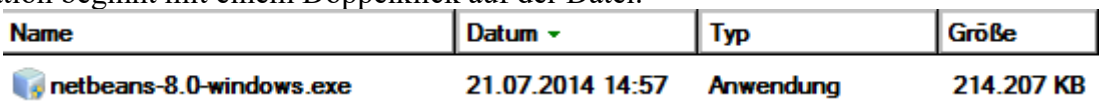
Die Java EE-Version enthält die genutzten Server, so dass auf die „All“-Variante verzichtet werden kann. Diese benötigt typischerweise mehr Arbeitsspeicher und ist langsamer. Wer mit weiteren Möglichkeiten von NetBeans z. B. mit im Umfeld von HTML5 experimentieren möchte, kann auch diese Variante nutzen. Man beachte oben im folgenden Bild die Auswahlmöglichkeiten für die Sprache und das Betriebssystem im rechten oberen Bereich, wobei „deutsch“ manchmal bei sehr aktuellen Versionen nicht zur Verfügung steht.



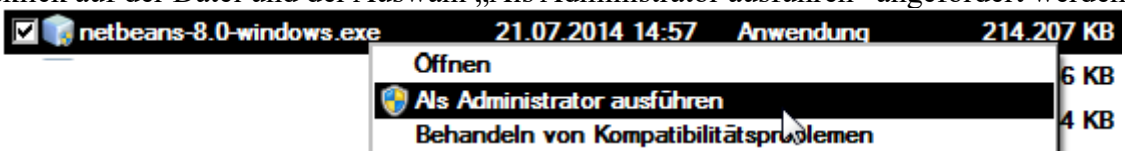
Danach kann der Download starten.



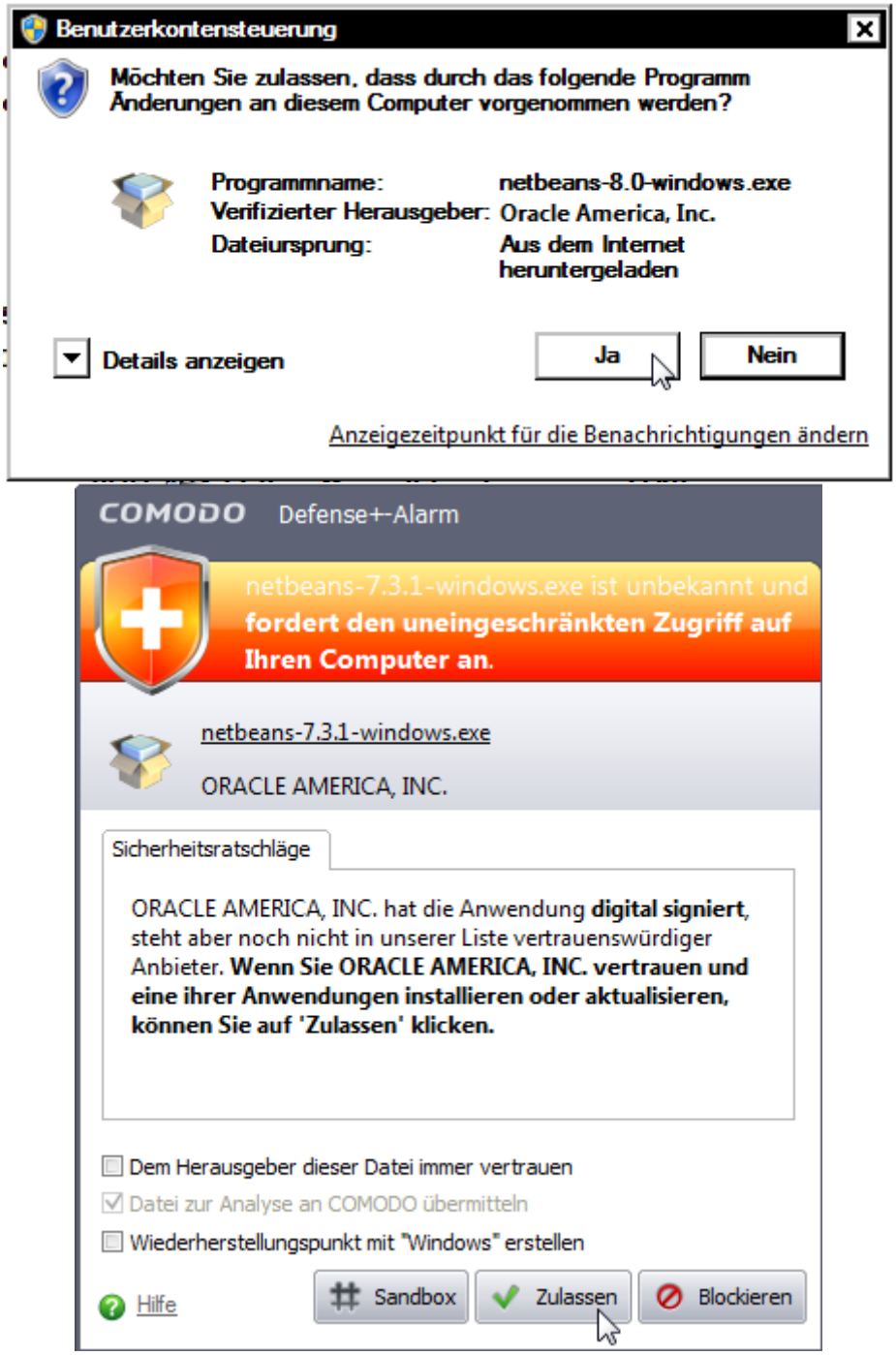
Die Installationsdatei kann in einem beliebigen Ordner mit Lese- und Schreibrechten liegen. Die Installation beginnt mit einem Doppelklick auf der Datei.



Generell nicht notwendig aber sinnvoll ist die Installation mit Administrationsrechten, die über einen Rechtsklick auf der Datei und der Auswahl „Als Administrator ausführen“ angefordert werden.

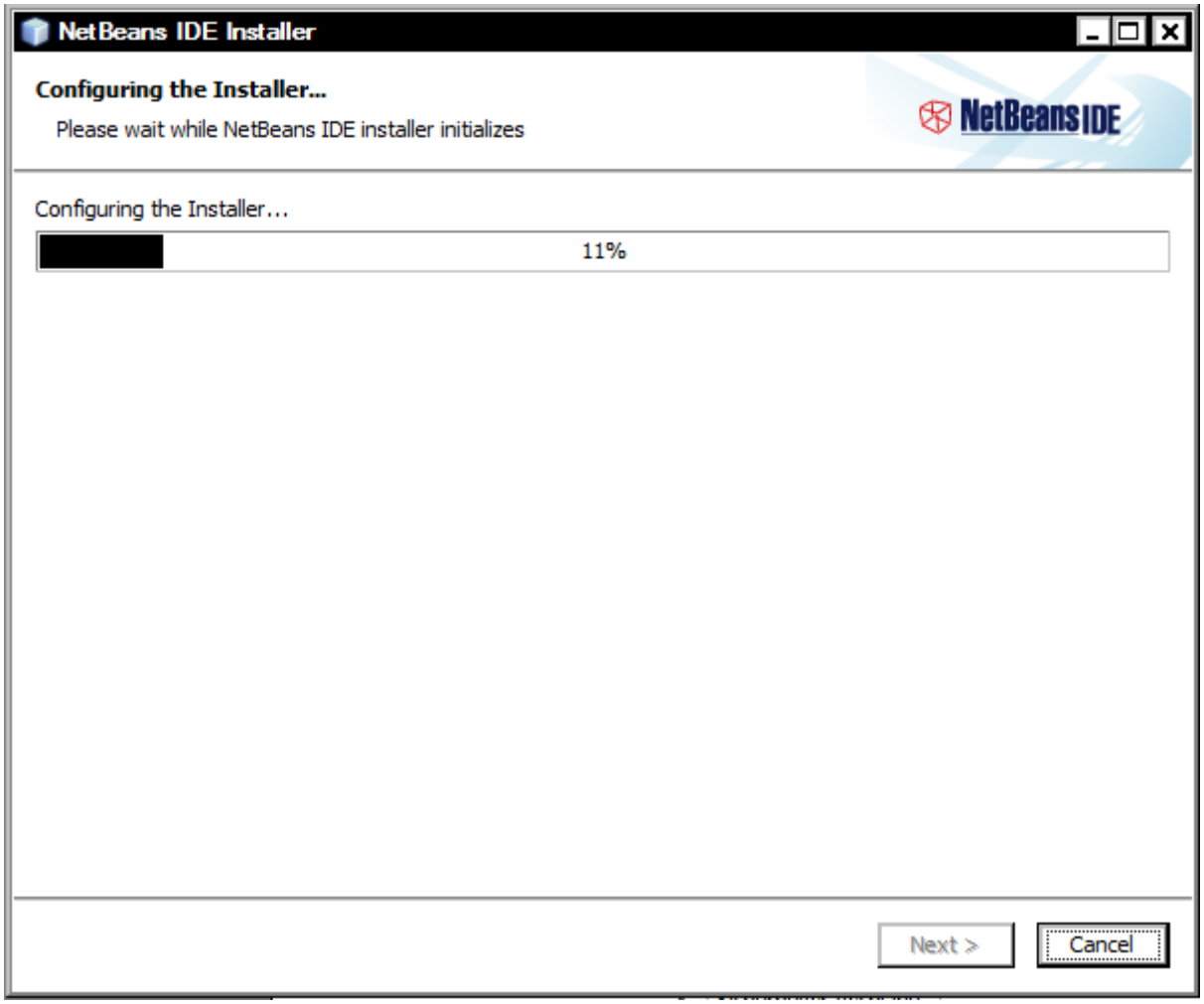


Eventuell muss die Installation bei der genutzten Firewall, wie im Beispiel, erlaubt werden. Alle diesbezüglich vorzunehmenden Änderungen sind in der Verantwortung des Installierenden.

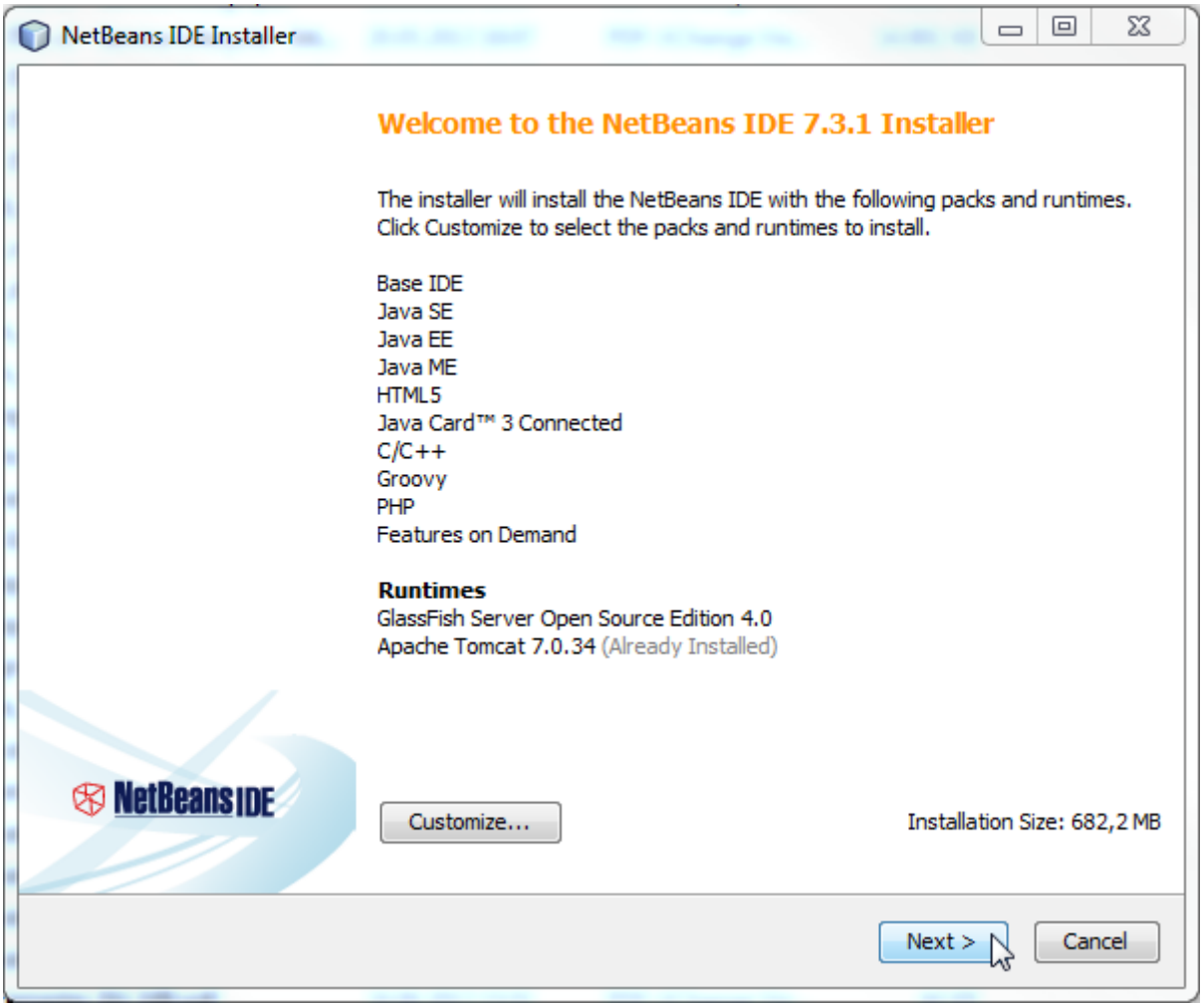

Benutzerkontensteuerung
Möchten Sie zulassen, dass durch das folgende Programm Änderungen an diesem Computer vorgenommen werden?
Programmname: netbeans-8.0-windows.exe
Verifizierter Herausgeber: Oracle America, Inc.
Dateiursprung: Aus dem Internet heruntergeladen
Details anzeigen
Ja
Nein
Anzeigezeitpunkt für die Benachrichtigungen ändern
COMODO Defense+-Alarm
netbeans-7.3.1-windows.exe ist unbekannt und fordert den uneingeschränkten Zugriff auf Ihren Computer an.
netbeans-7.3.1-windows.exe
ORACLE AMERICA, INC.
Sicherheitsratschläge
ORACLE AMERICA, INC. hat die Anwendung digital signiert, steht aber noch nicht in unserer Liste vertrauenswürdiger Anbieter. Wenn Sie ORACLE AMERICA, INC. vertrauen und eine ihrer Anwendungen installieren oder aktualisieren, können Sie auf 'Zulassen' klicken.
Dem Herausgeber dieser Datei immer vertrauen
Datei zur Analyse an COMODO übermitteln
Wiederherstellungspunkt mit "Windows" erstellen
Hilfe
Sandbox
Zulassen
Blockieren



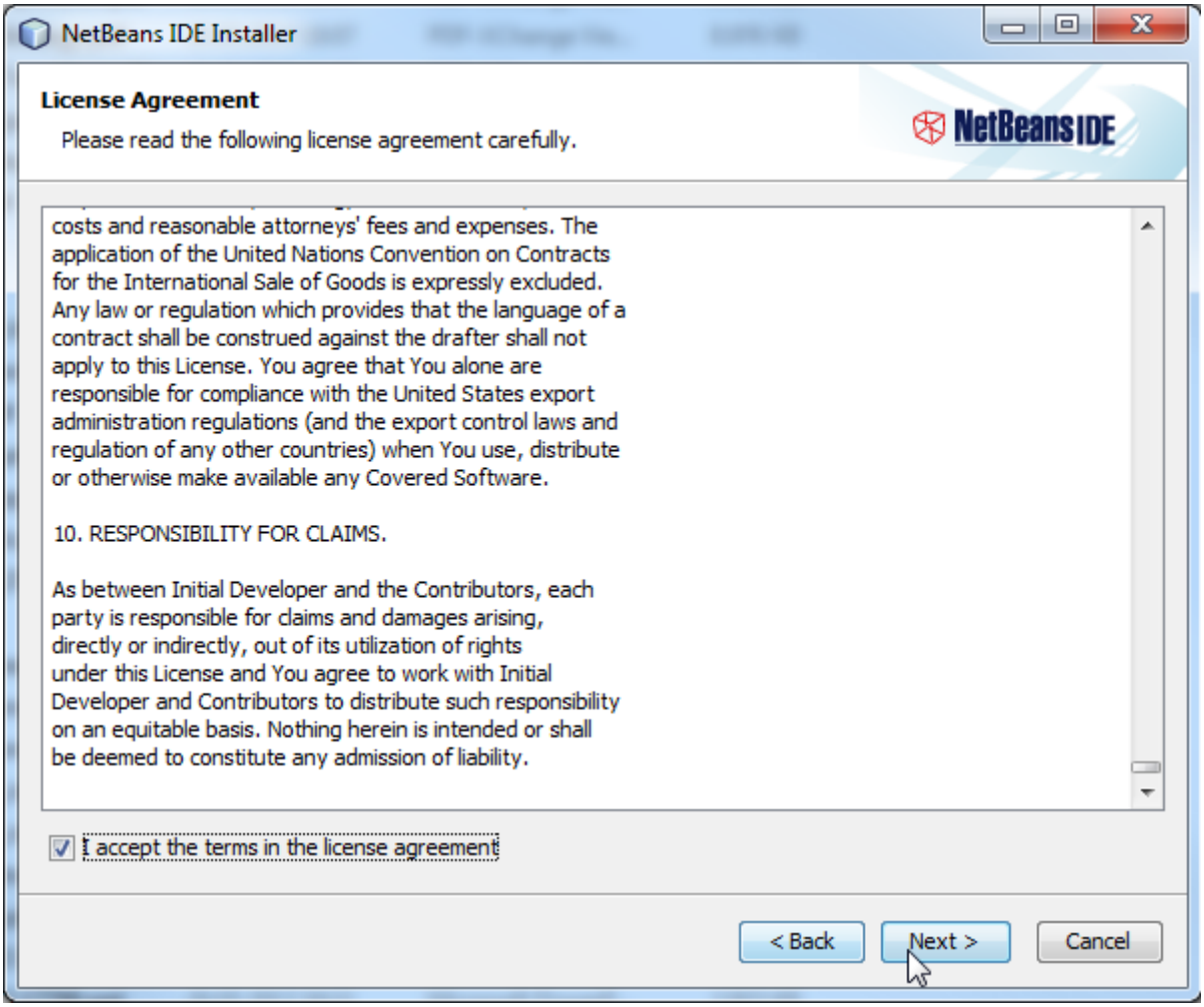
Der Konfigurationsvorgang am Start kann etwas dauern.



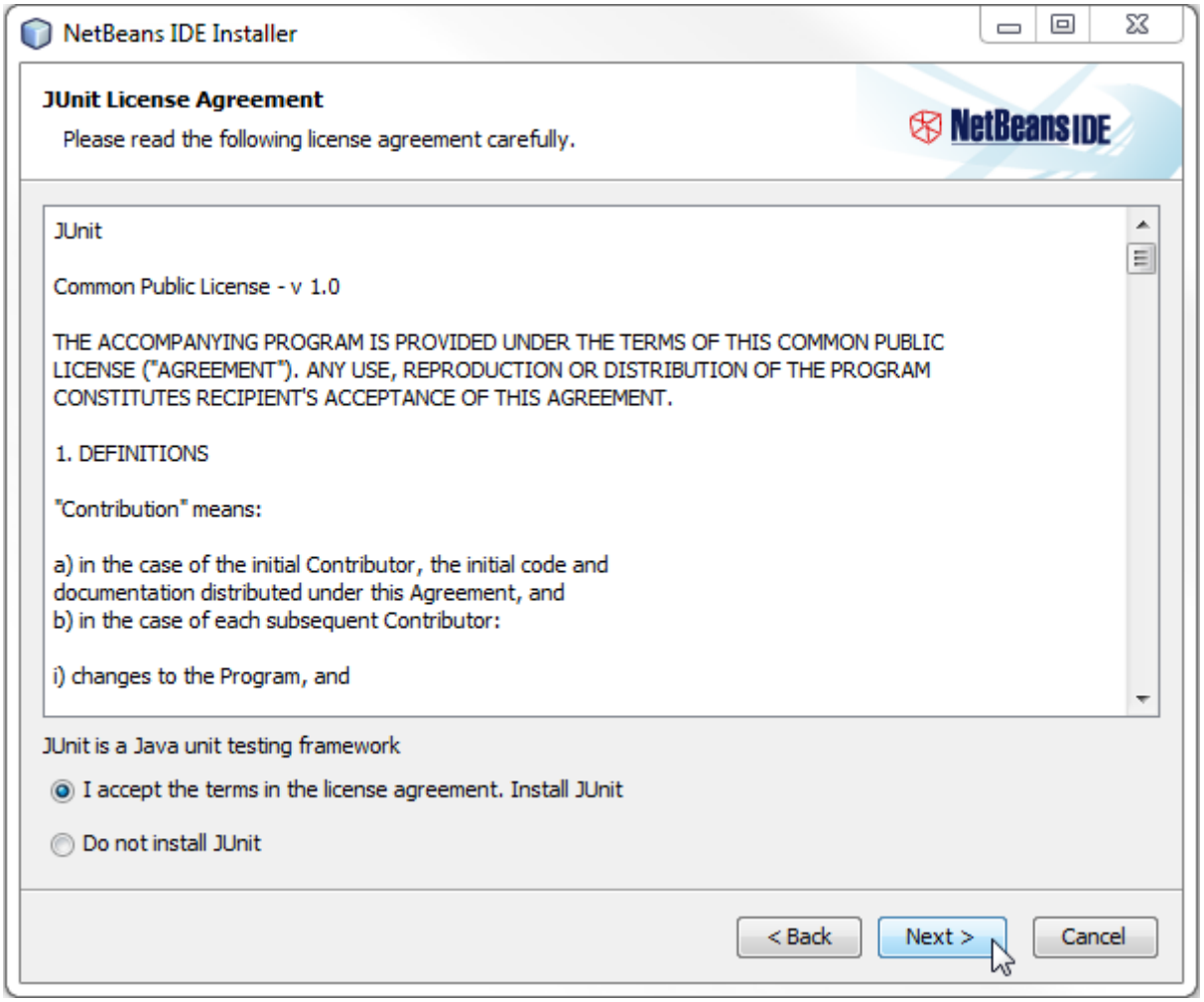
Am Start kann man die gewünschten Pakete und Applikationsserver über „Costomize“ auswählen. In den Veranstaltungen wird meist der GlassFish genutzt. Man kann aber auch beide Server installieren, da diese später von Hand in NetBeans gestartet werden. Werden EJBs genutzt, wird dies in NetBeans nur direkt mit GlassFish unterstützt, da für Tomcat zunächst z. B. Apache TomEE (http://openejb.apache.org/apache-tomee.html) nachinstalliert werden muss. Hier wird einfach „Next >“ geklickt. Hier wird die Variante gezeigt, bei der ein GlassFish-Server direkt mitinstalliert wird, eine Alternative dazu wird im nachfolgenden Unterkapitel mit der Installation eines aktuelleren GlassFish-Servers gezeigt. In einer weiteren Variante kann man die gewünschten Server mit GlassFish und Apache Tomcat auswählen, wobei beide Server parallel nutzbar sind.



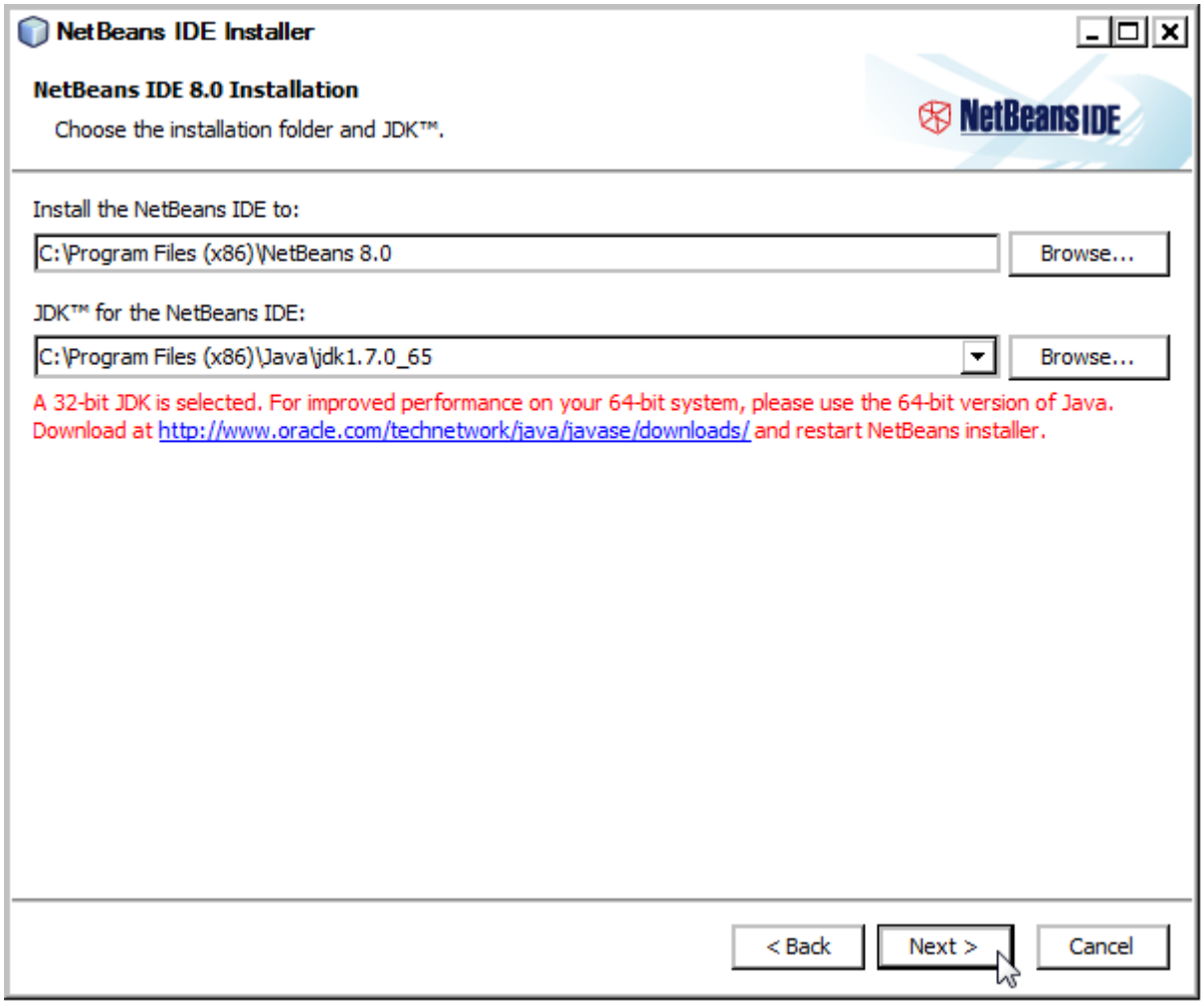
Die Lizenz wird durchgelesen und muss zur Installation angenommen werden,



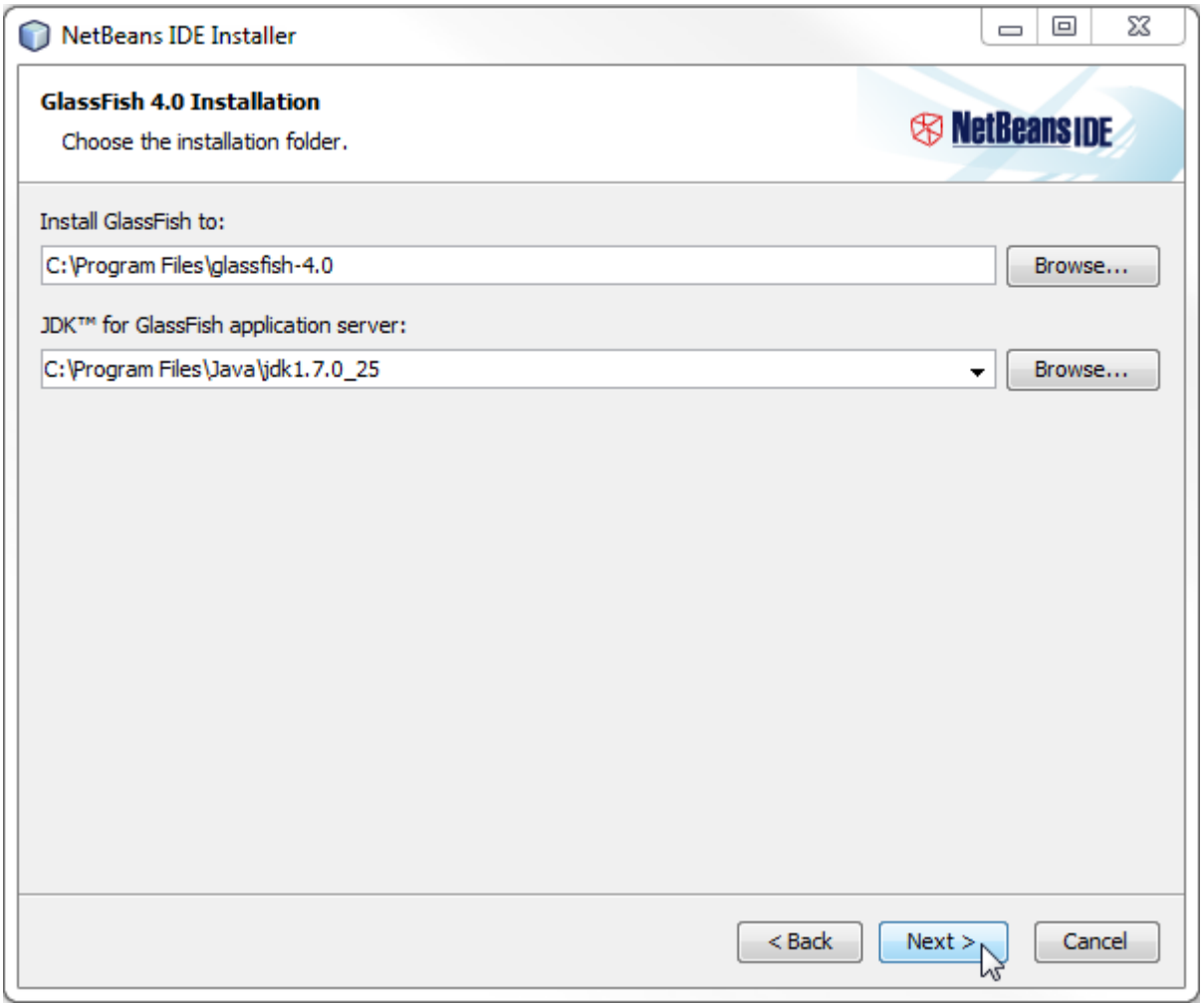
Die Lizenzbedingungen von JUnit werden auch durchgelesen und sollten akzeptiert werden, da JUnit ein zentrales Hilfsmittel für Tests ist.



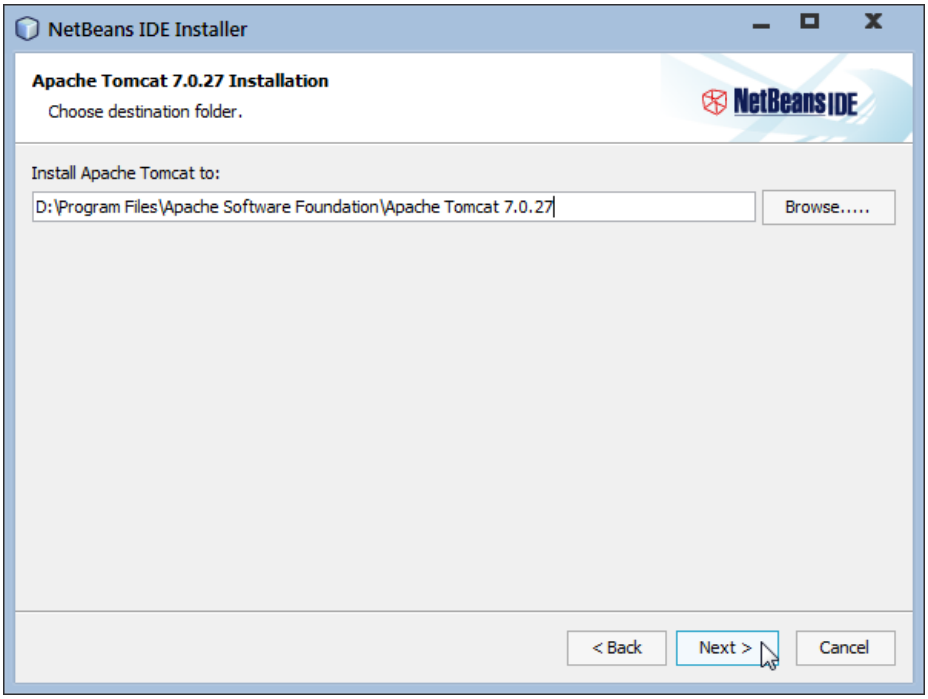
Der Installationspfad und ein installiertes JDK werden gewählt. Meist können die Einstellungen so übernommen werden. Der Hinweis auf eine 32-bit JDK-Version ist wichtig, kann aber ignoriert werden, wenn man z. B. mit verschiedenen Umgebungen Java nutzt. Es ist fast immer der Fall, dass eine 32-bit-Version auf allen und mit fast allen 64-bit-Systemen zusammen läuft.



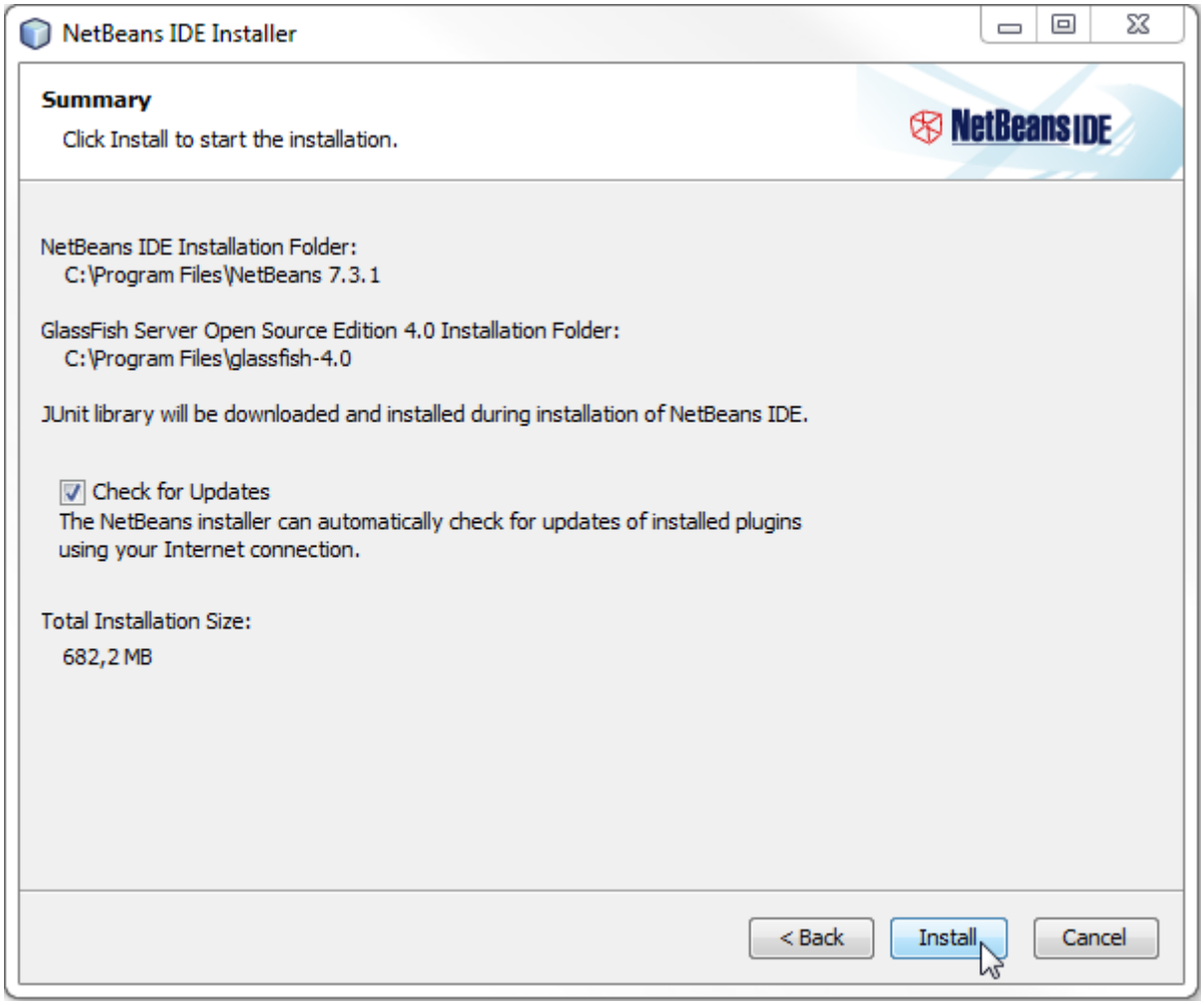
Der Ordner für dan Installationsordner von GlassFish wird gewählt.



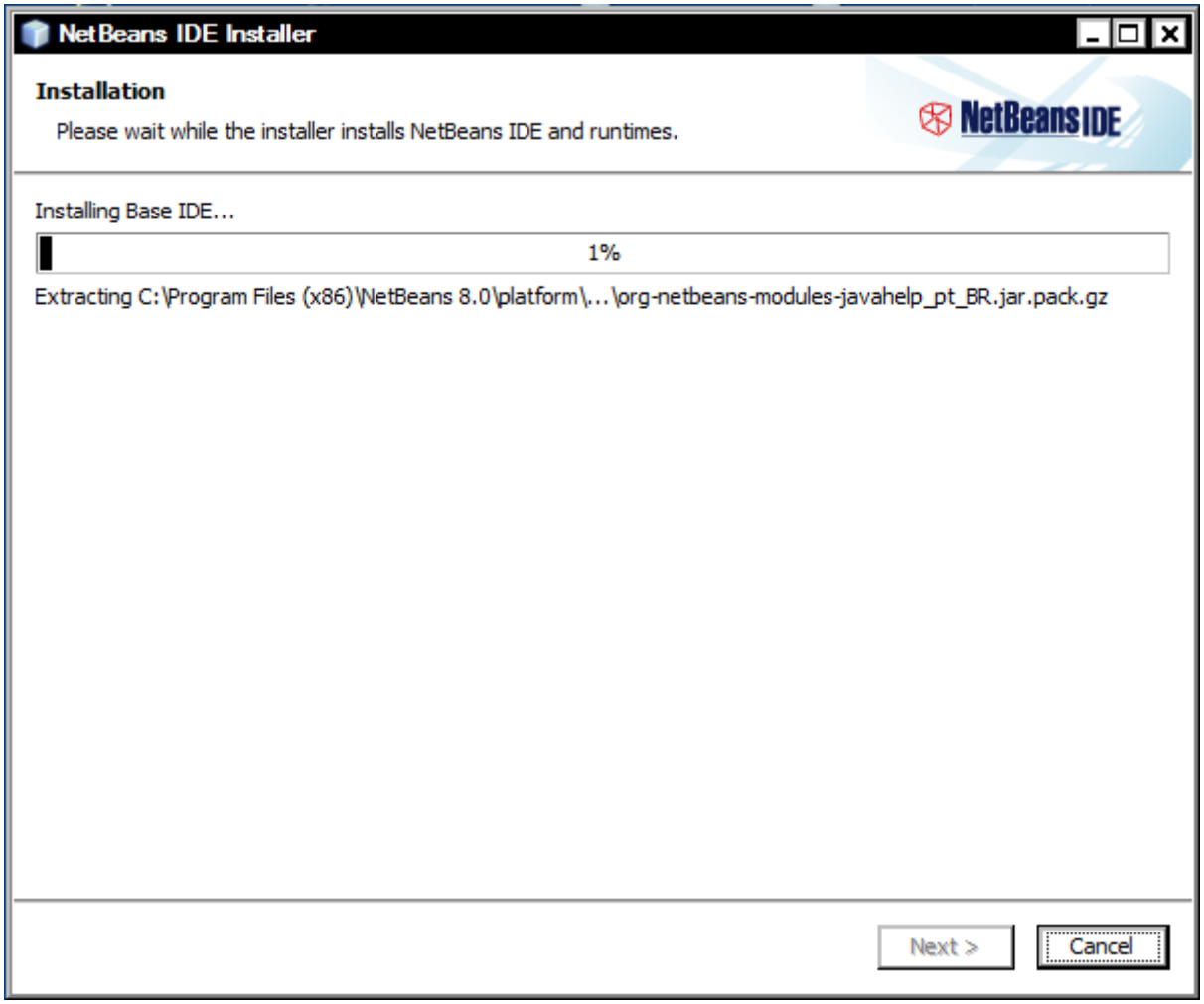
Der Ordner für den Apache Server wird gewählt, insofern TomCat installiert wird. Bei neueren NetBeans-Versionen wird dieser Schritt gar nicht angeboten und muss ignoriert werden.



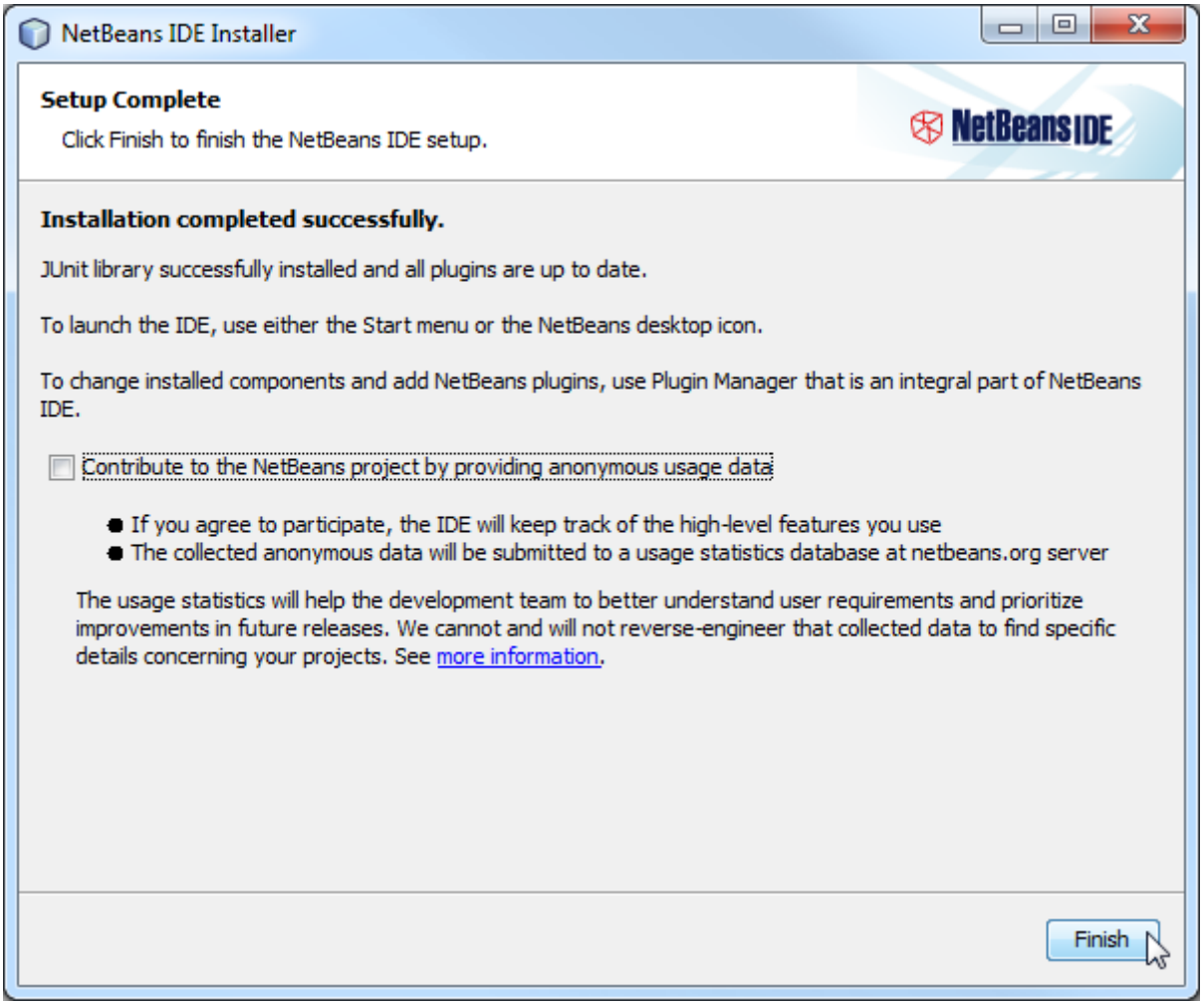
Danach kann die Installation gestartet werden. Man beachte, das JUnit aus dem Internet nachgeladen wird.



Die Installation kann eine sehr lange Zeit in Anspruch nehmen.

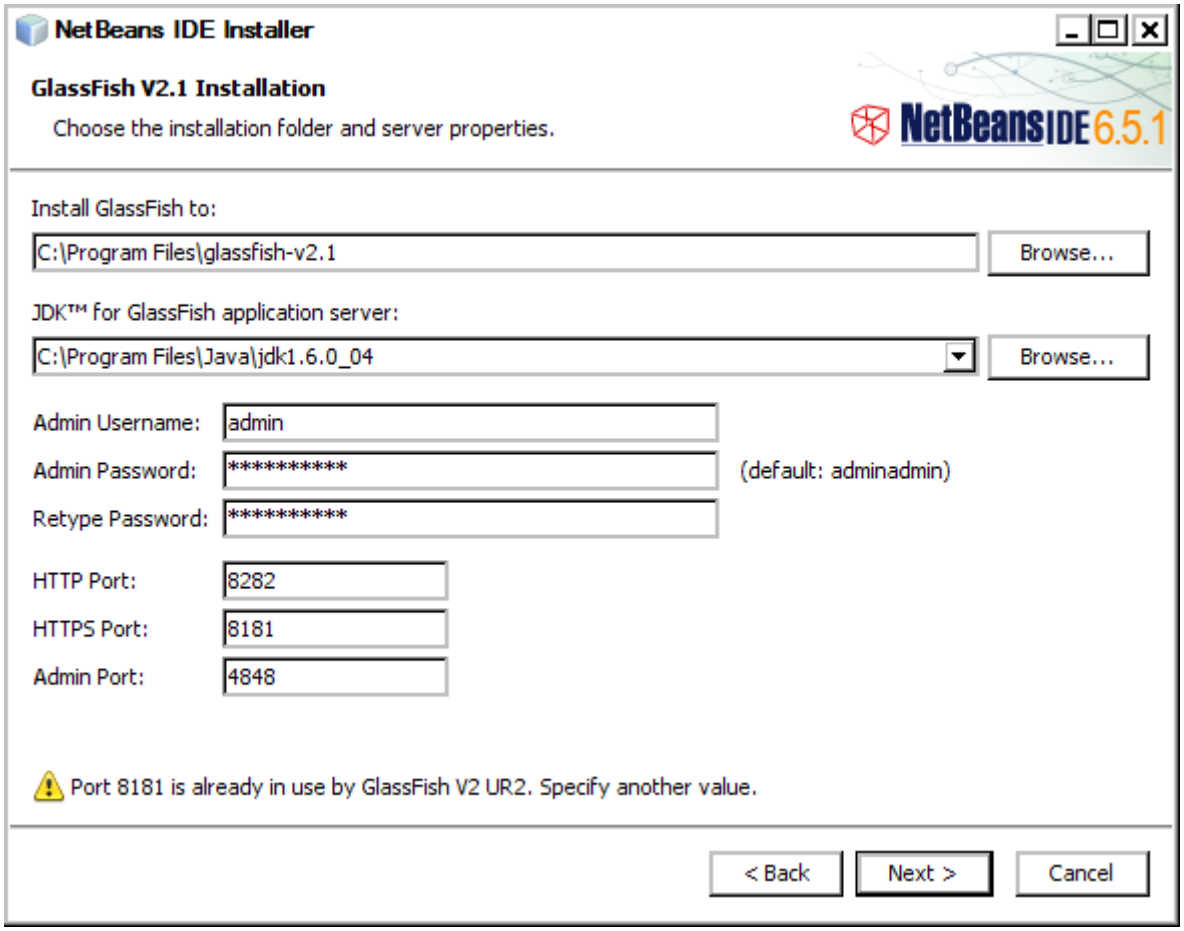


Zum Abschluss kann man entscheiden, ob anonyme Nutzerdaten weitergeleitet werden sollen.

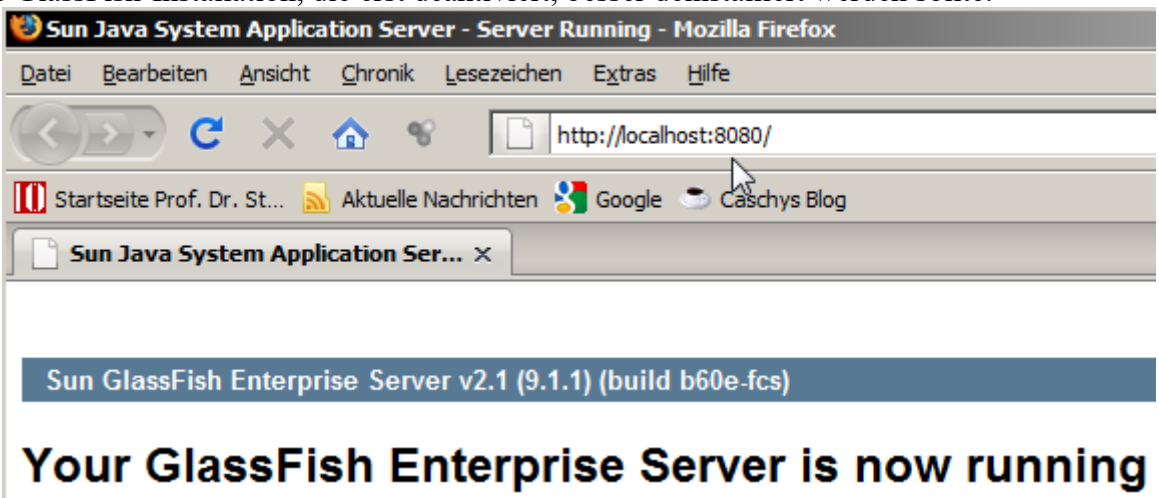


Die folgenden Notizen wurden für ältere NetBeans-Versionen erstellt und müssen nach erfolgreich abgeschlossener Installation nicht beachtet werden. Der Inhalt könnte aber für mögliche Probleme und einige wenige Hintergründe interessant sein.

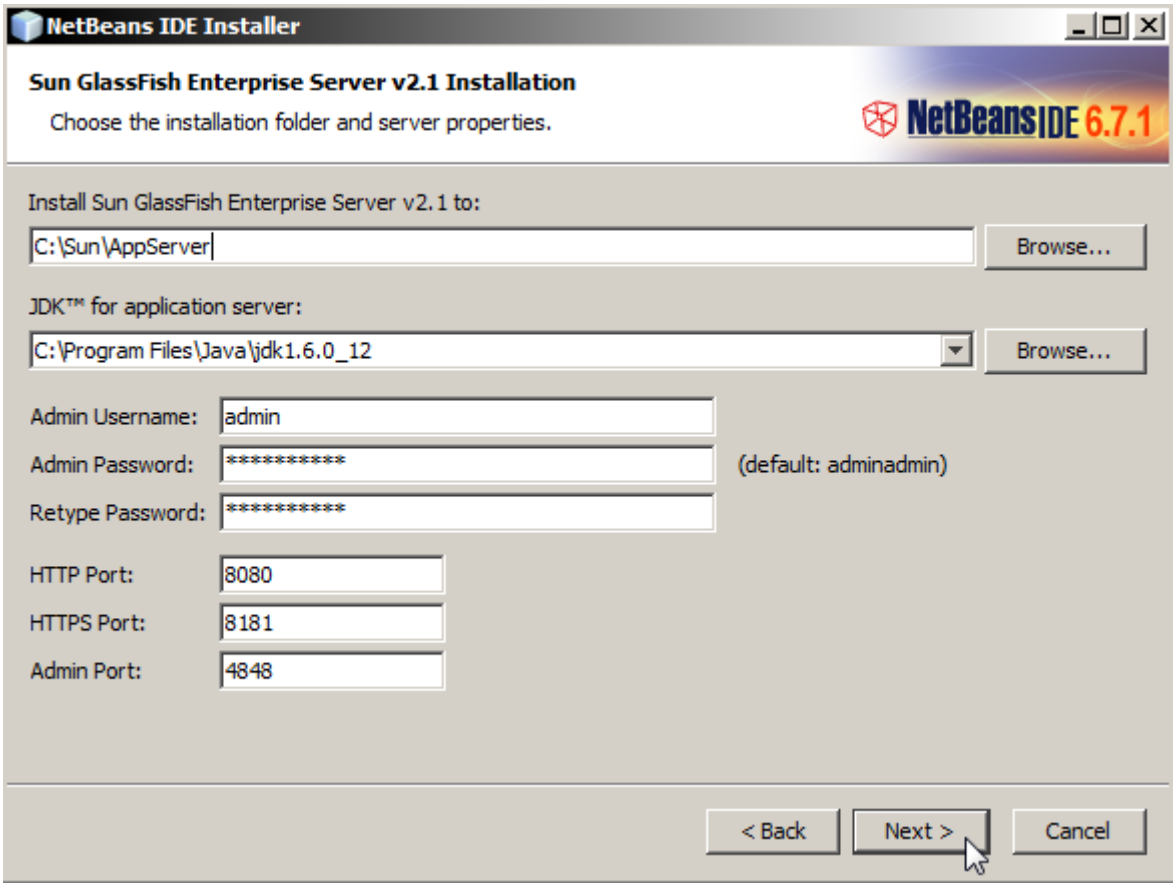
Die folgenden Einstellungen sind zentral für die weitere Arbeit und sollten genau analysiert werden. Das folgende Fenster zeigt als kritisches Beispiel einer älteren Installation, dass die Standardinstallation nicht möglich ist. Der HTTP Port ist üblicherweise 8080, HTTPS Port 8181 und der Admin Port von GlassFish, über den der Server konfiguriert werden kann, 4848. Das Ausrufungszeichen unten zeigt an, dass die üblichen Einstellungen nicht wählbar sind.



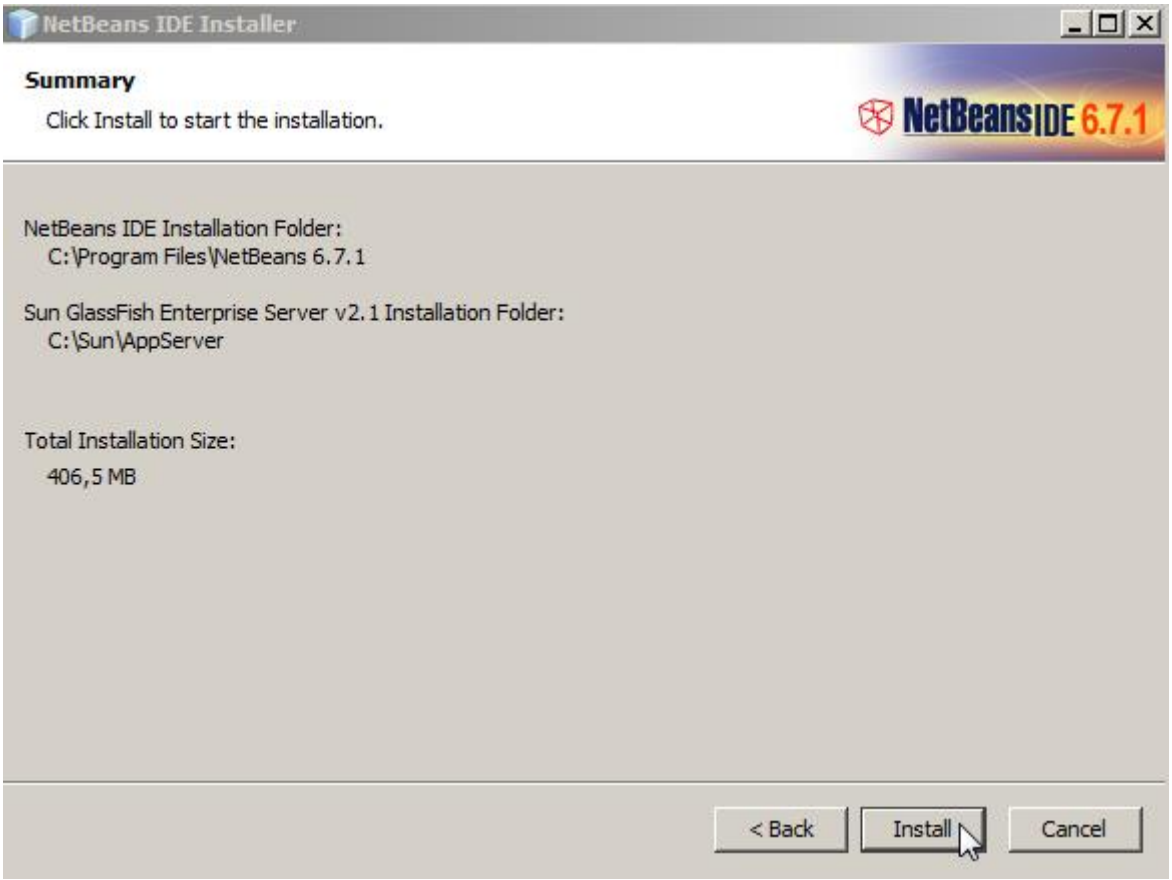
Man kann mit den vorgestellten Einstellungen arbeiten, sollte aber darüber nachdenken, warum z. B. der Port 8080 belegt ist. Eine Möglichkeit direkt mit dem Port „Kontakt aufzunehmen“ besteht über einen Browser, der localhost:8080 abfragt. Insofern der Prozess, der diesen Port nutzt, direkt antworten kann, erhält man weitere Informationen über die Port-Nutzung. Im Beispiel hier läuft eine ältere GlassFish-Installation, die erst deaktiviert, besser deinstalliert werden sollte.



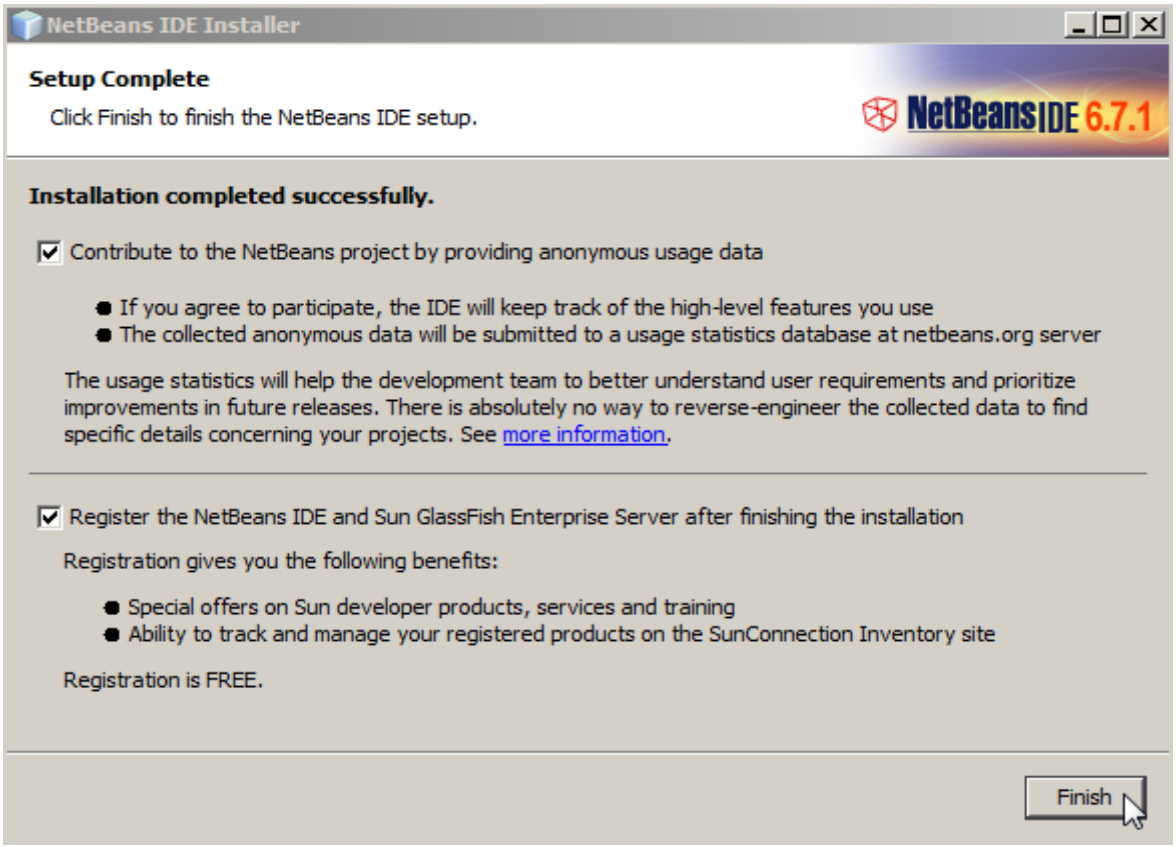
Eine Installation kann über „Cancel“ abgebrochen werden. Nach dem Aufräumen und der Wiederholung der Installationsschritte kommt man dann zu folgendem Fenster, mit leicht geänderten Einträgen. Sehr wichtig ist es, dass man sich den Administrator und das Passwort für den Server merkt. Die Default-Einstellungen könnten kritisch werden, falls der Rechner immer läuft und die Firewall Zugriffe von außerhalb auf den Port 4848 nicht blockt. Trotzdem wird angeraten, diese Werte zunächst zu übernehmen.



Danach kann die eigentliche Installation durchgeführt werden.

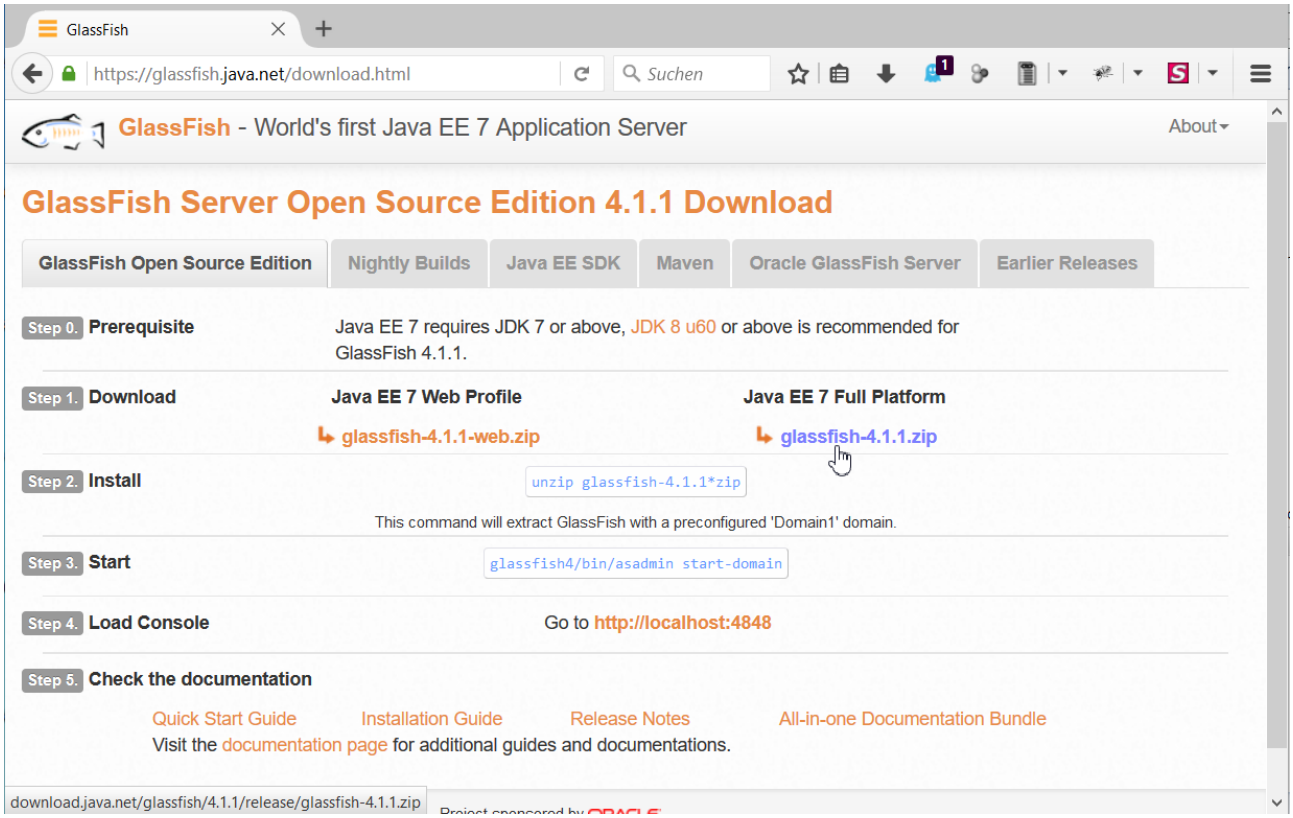

NetBeans IDE Installer
Summary
Click Install to start the installation.
NetBeans IDE 6.7.1
NetBeans IDE Installation Folder:
C:\Program Files\NetBeans 6.7.1
Sun GlassFish Enterprise Server v2.1 Installation Folder:
C:\Sun\AppServer
Total Installation Size:
406,5 MB
< Back
Install
Cancel

Für die folgenden zwei Bitten von Sun muss jeder selbst entscheiden, wie er sie beantwortet.

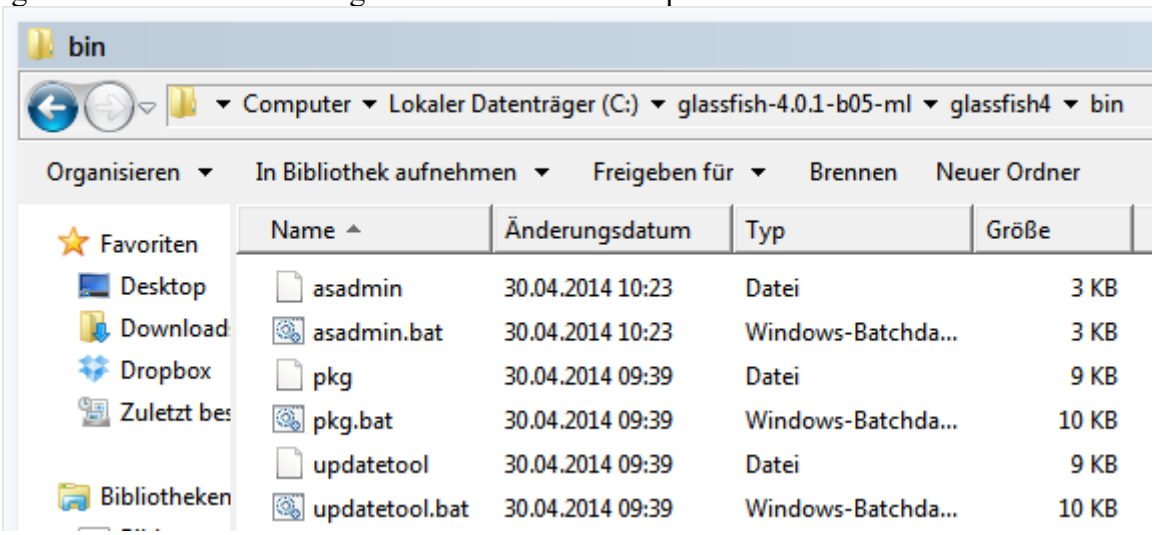


2.2 Installation eines eigenen GlassFish-Servers

Da GlassFish nicht mehr im Fokus der Entwicklungen bei Oracle steht, finden Aktualisieren seit einiger Zeit spärlicher statt. Trotzdem bleibt GlassFish die Referenzimplementierung für JEE, so dass hier alle aktuellen Technologien vereinigt werden. Um dann aber wirklich den aktuellsten Stand der einzelnen Teiltechnologien, wie JSF, CDI, JEE, zu nutzen, muss man häufig Beta-Versionen betrachten, die das Experimentieren mit den neusten Ständen ermöglichen. Die Installation geht von einer aktuellen JDK-Installation mit gesetzten System-Variablen JAVA_HOME und JDK_HOME aus.
Aktuelle Versionen von GlassFish können unter https://glassfish.java.net/download.html geladen werden. Dazu wird auf die aktuellste Versionsnummer bei der „Full Platform“ geklickt.



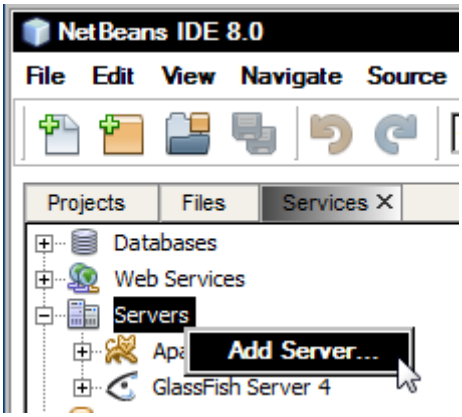
Die eigentliche Installation erfolgt einfach durch das Auspacken des Verzeichnisses.



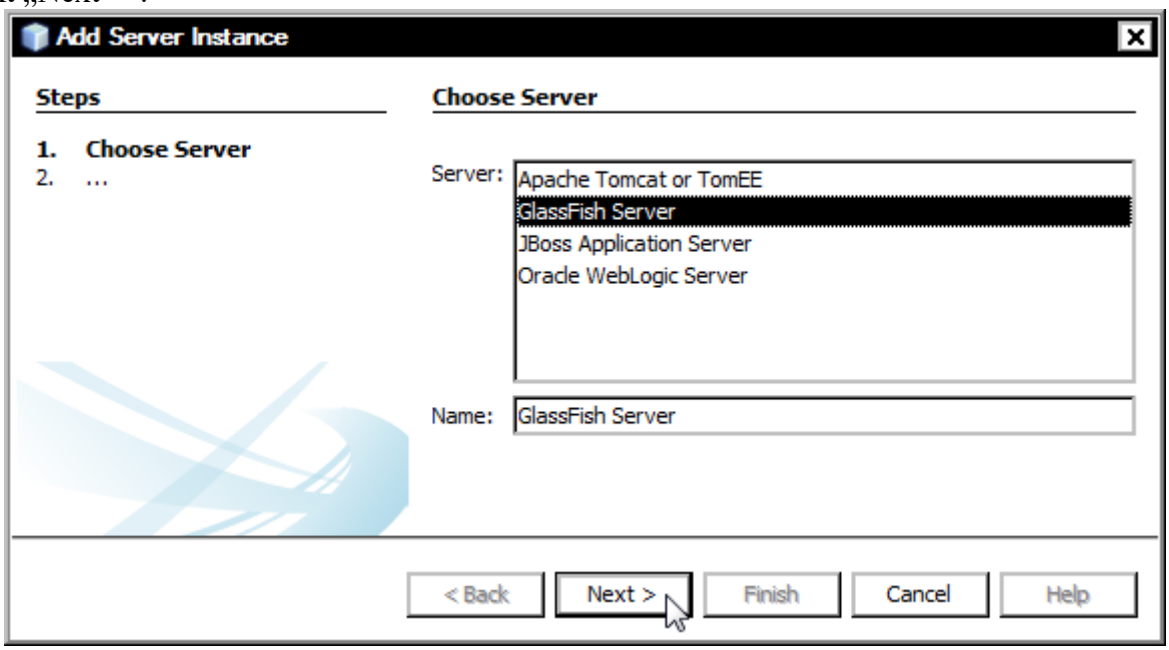
2.3 Einbindung eines neuen GlassFish-Servers in NetBeans

Falls man noch nicht mit NetBeans gearbeitet hat, sollte man zunächst die folgenden Kapitel durcharbeiten, bevor man die lokale GlassFish-Installation in NetBeans einbindet.
Unter dem Reiter Services sind unter „Servers“ die bisher bekannten Server eingetragen, bei denen der neue Server jetzt ergänzt werden soll. Dazu wird ein Rechtsklick auf „Servers“ gemacht und „Add

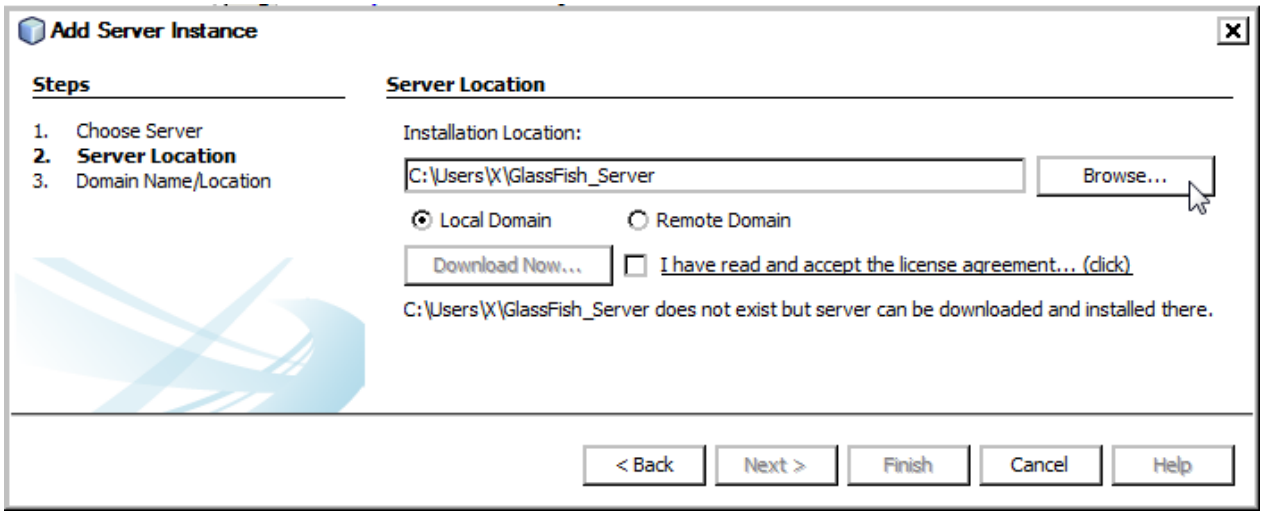
Server…“ ausgewählt.



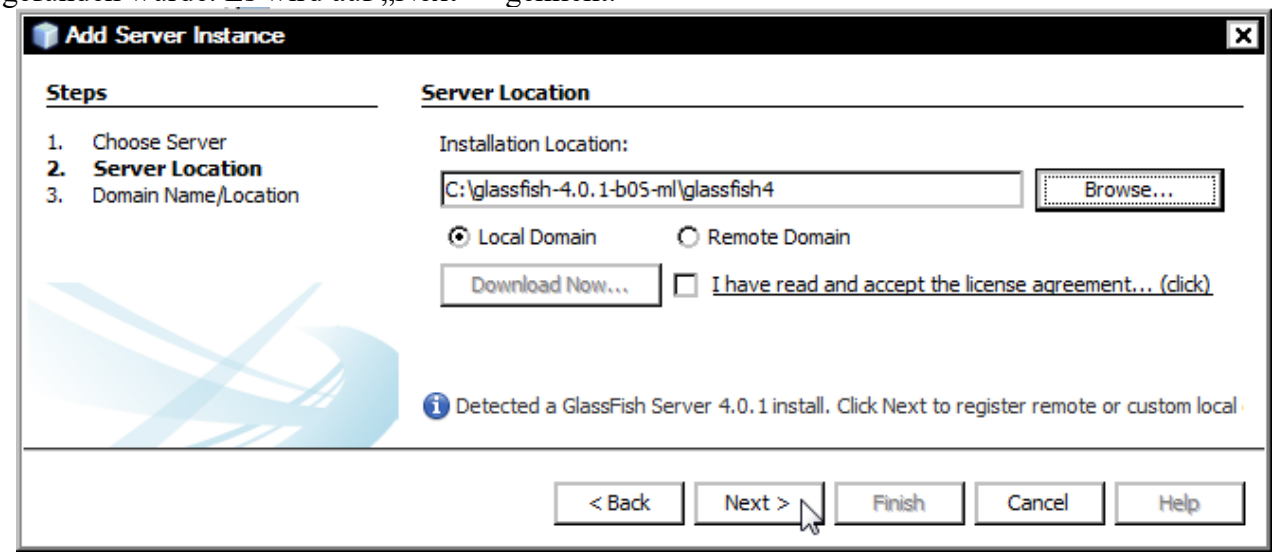
Man wählt oben den passenden Server, hier „GlassFish Server“, gibt unten einen Namen ein und klickt „Next >“.



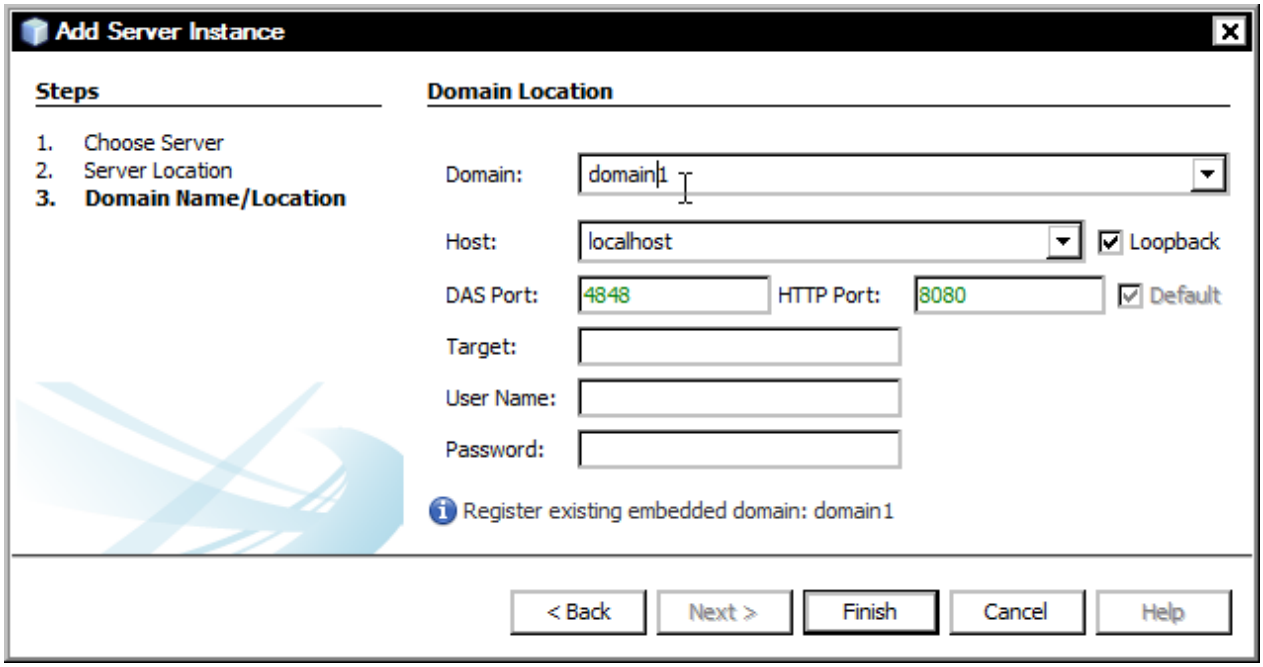
Nun muss unter „Browse…“ der Ordner der GlassFish-Installation ausgewählt werden. da es mehrere Ordner mit ähnlichen Namen gibt, ist der passende Ortdner wichtig, wobei unten angegeben wird, ob ein Server gefunden wurde.



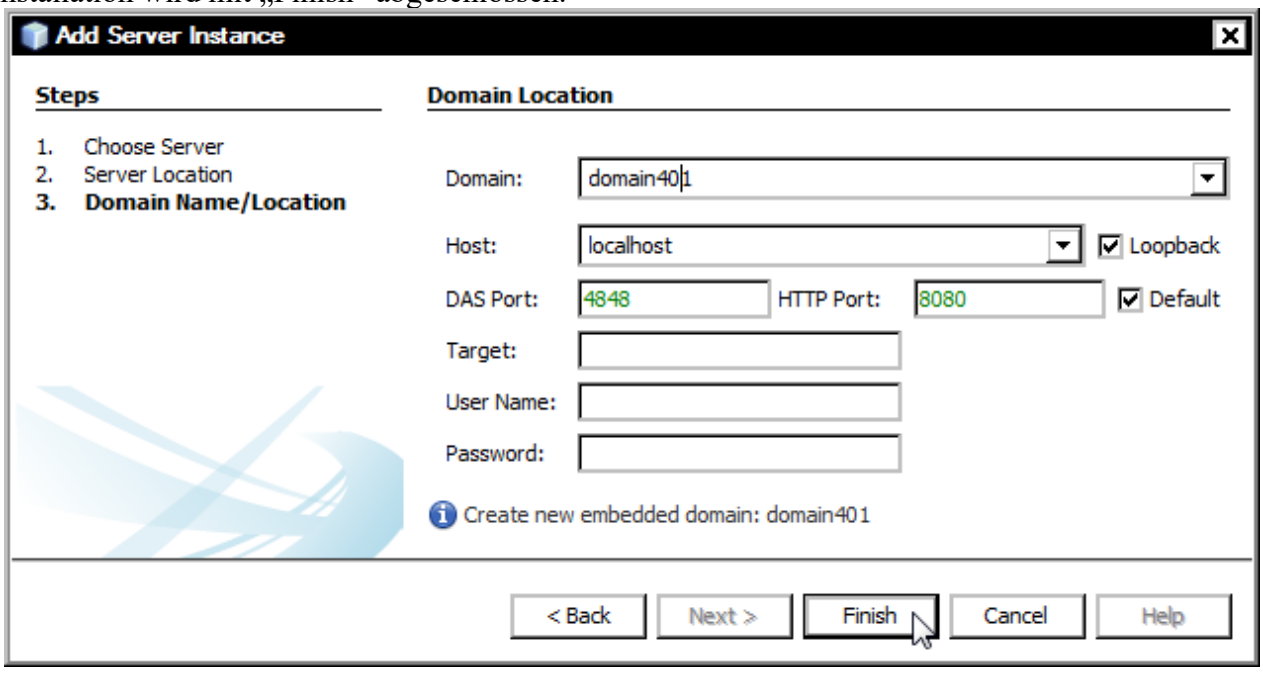
Das passende Verzeichnis hat den Namen „glassfish4“, die untere Meldung zeigt auch, dass der Server gefunden wurde. Es wird auf „Next >“ geklickt.



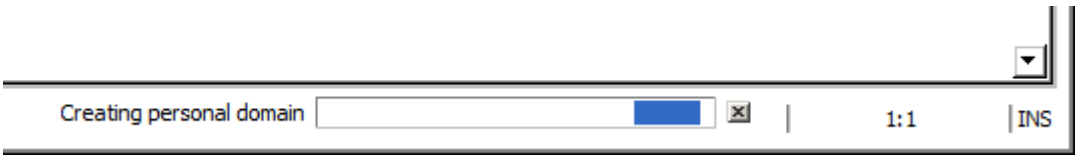
Das folgende Fenster zeigt, dass die abgegebene Domain bereits existiert, sie sollte dann umbenannt werden. Man kann weiterhin einen Nutzer (User Name) und ein Passwort (Password) angeben, mit dem der Server verwaltet werden kann. Zum Experimentieren können alle Felder leer gelassen werden.



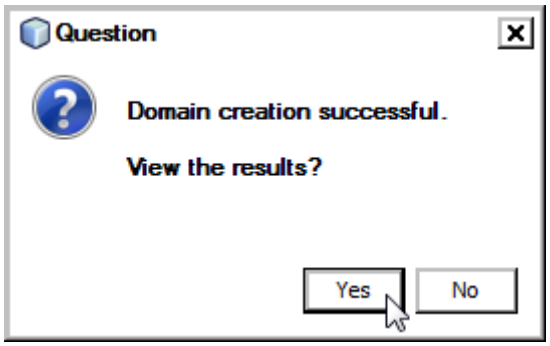
Man erkennt, dass nur der Domain-Name geändert wurde, was formal auch nicht notwendig ist. Die Installation wird mit „Finish“ abgeschlossen.



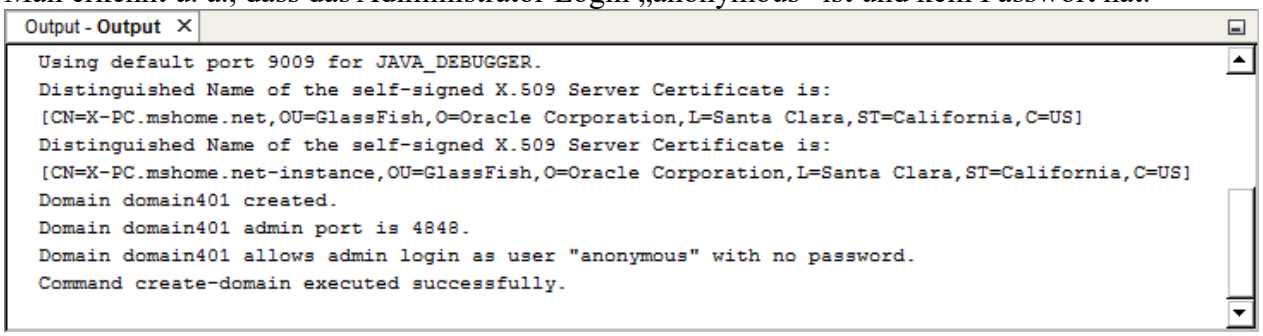
Der Aufbau der Domain kann etwas dauern.



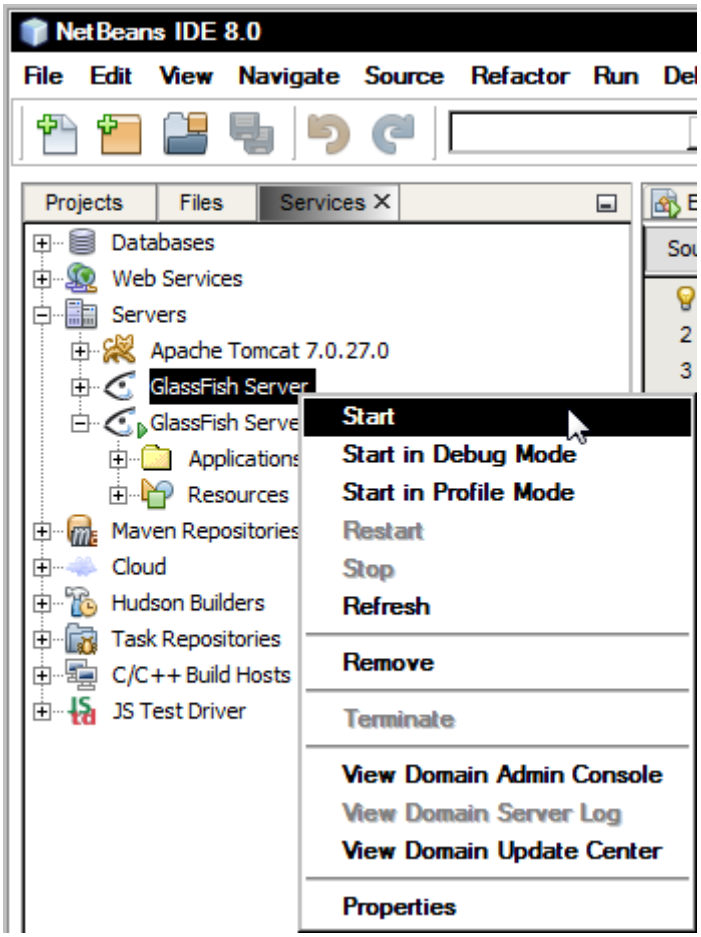
Man erhält eine Ergebnisnachricht und kann sich bei Interesse Details des Ergebnisses anschauen.



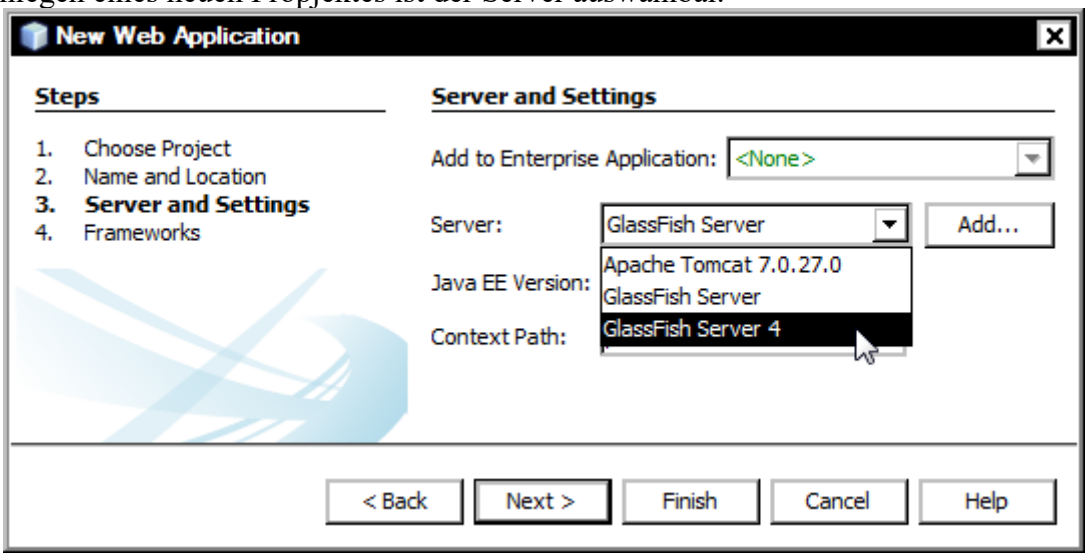
Man erkennt u. a., dass das Adiministrator Login „anonymous“ ist und kein Passwort hat.



Der neu installierte Server kann jetzt wie die anderen genutzt werden.

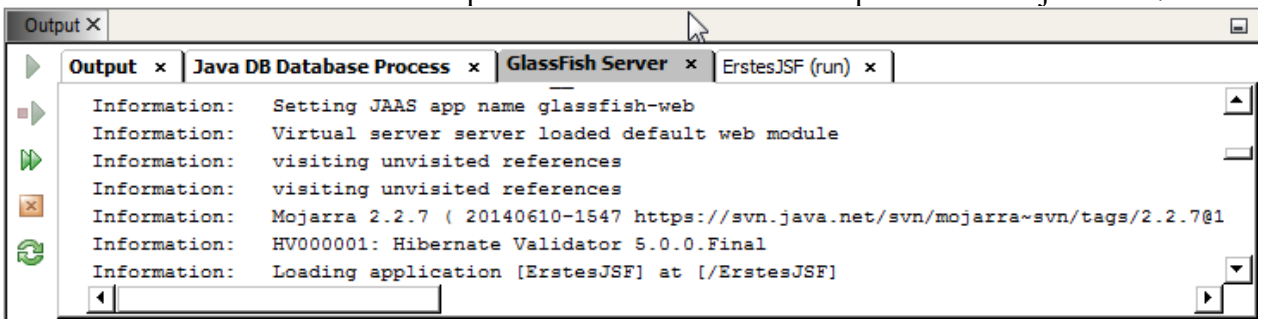


Beim Anlegen eines neuen Propjektes ist der Server auswählbar.

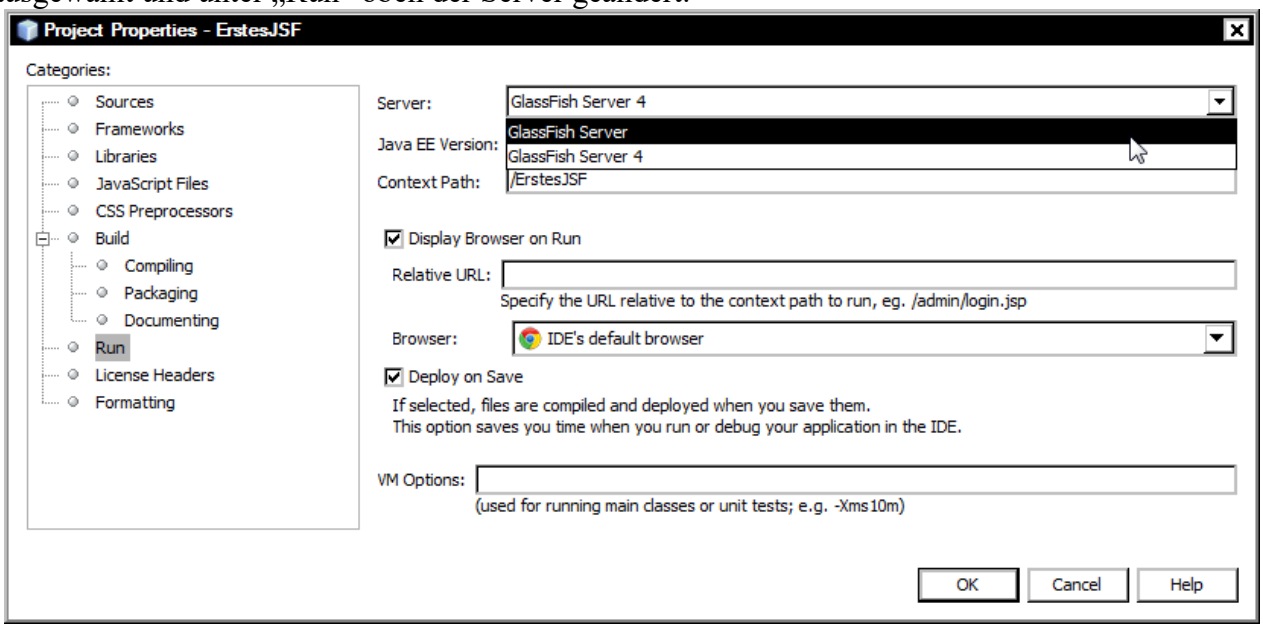


Der Server wird bei der ersten Ausführung eines Programmes genau wie die anderen Server gestartet. Im Protokoll unten im Reiter „GlassFish“ erhält man Startinformationen, die u. a. auch über die

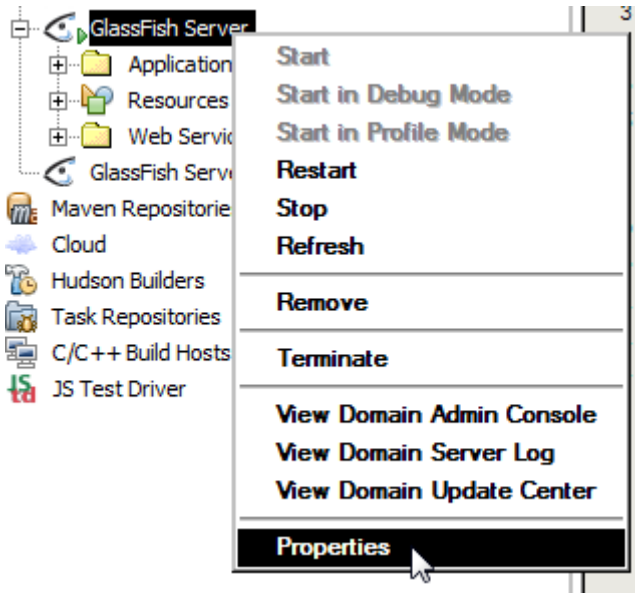
Versionen der verwendeten JEE-Komponenten informieren. Im Beispiel ist das Mojarra 2.2.7 für JSF.



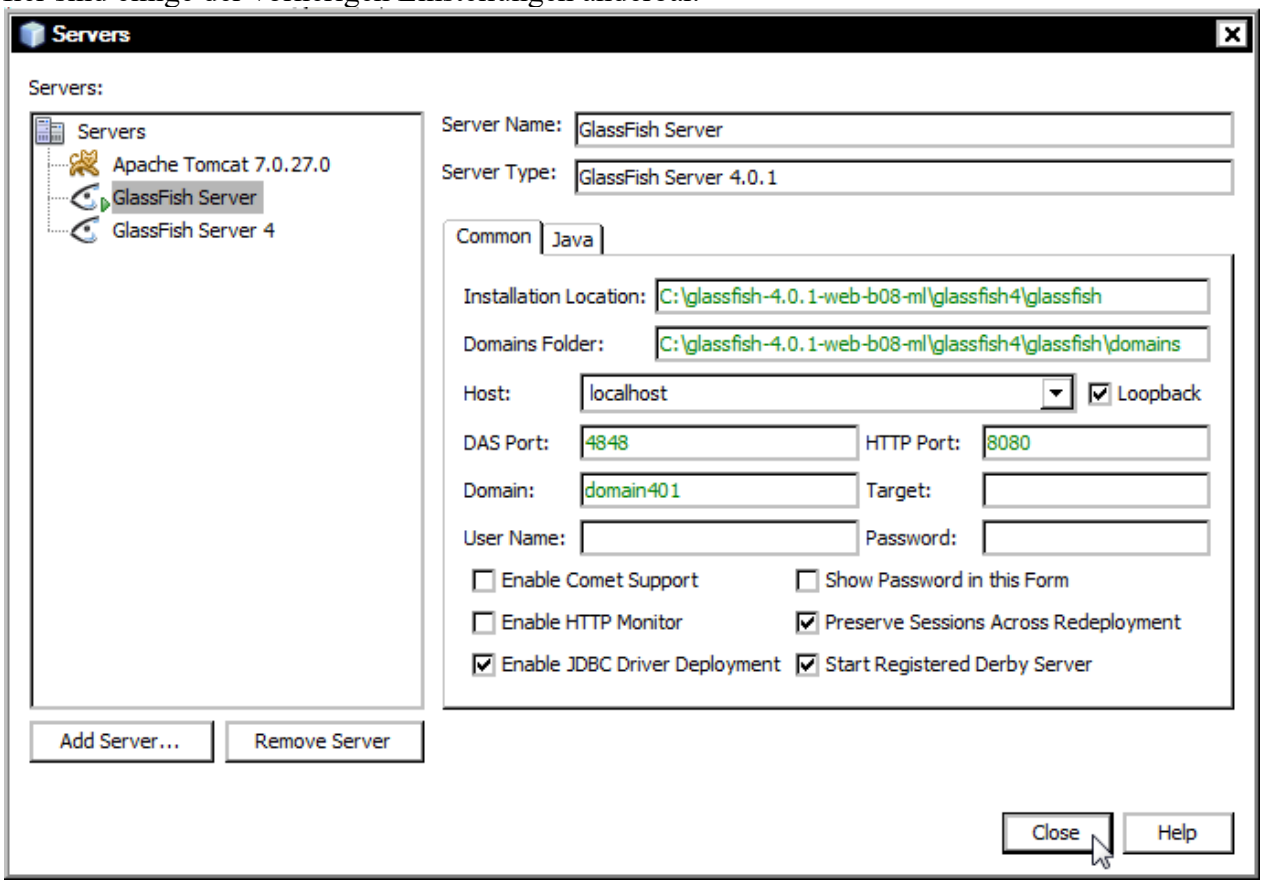
Möchte man später den genutzten Server ändern, wird auf das Projekt geklickt, „Properties“ ausgewählt und unter „Run“ oben der Server geändert.



Mit einem Rechtsklick auf dem Server, kann man auch die Einstellungen unter „Properties“ ändern.



Hier sind einige der vorherigen Einstellungen änderbar.



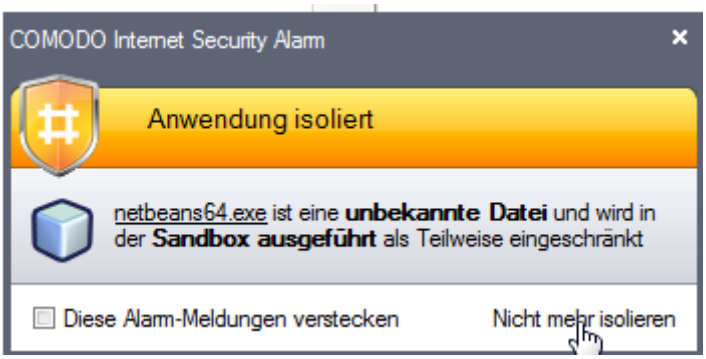
3 Erste Schritte mit NetBeans (NetBeans 8)

3.1 Erster Start

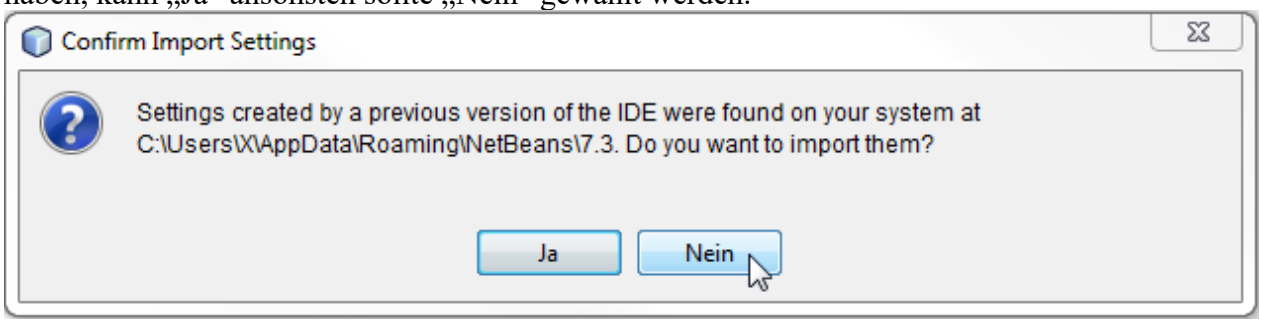
NetBeans wird zunächst gestartet.



Benötigte Sicherheitseinstellungen müssen individuelle, abhängig z. B. von der Firewall, vorgenommen werden.



Falls früher eine ältere Version von NetBeans auf dem Rechner installiert war, könnte am Anfang eine Frage der folgenden Form gestellt werden. Wenn man bereits erfolgreich mit NetBeans gearbeitet haben, kann „Ja“ ansonsten sollte „Nein“ gewählt werden.

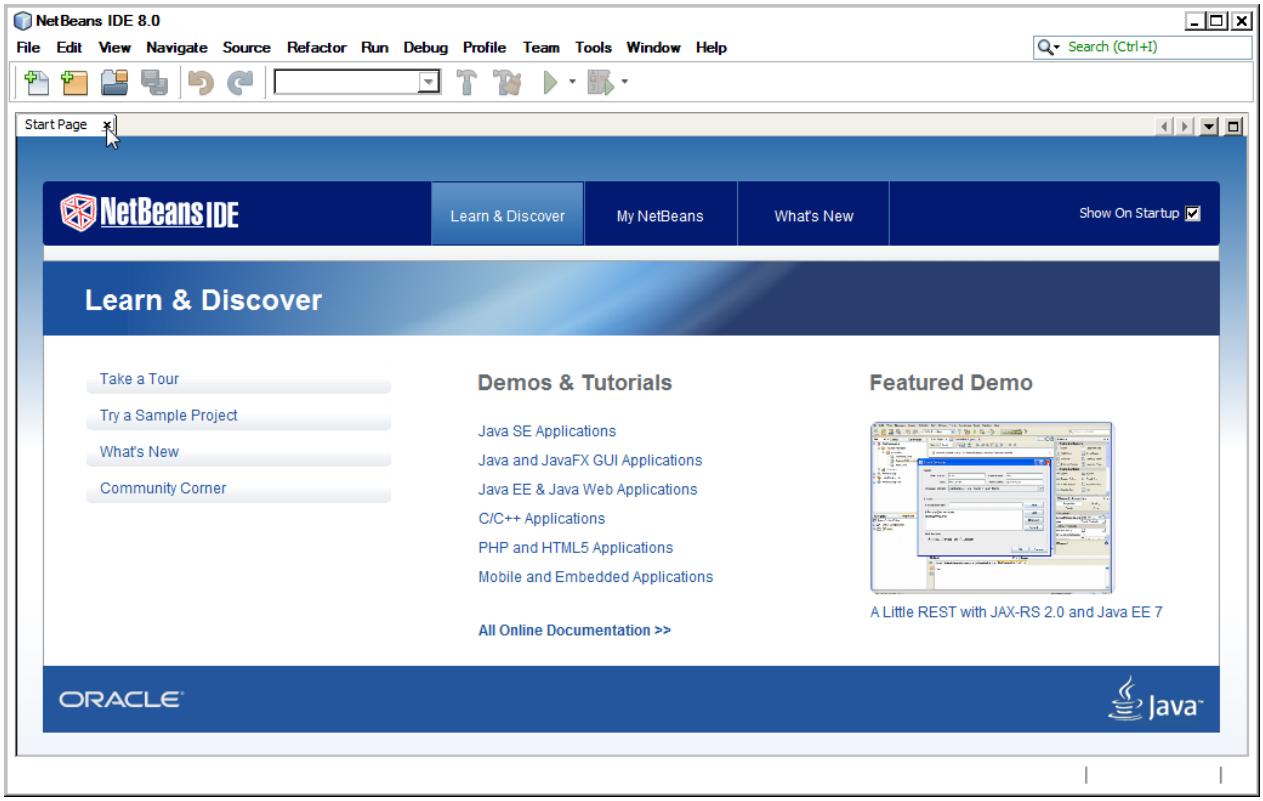


Wie es sich für ein gutes Java-Programm gehört, dauert der Programmstart etwas; wenn alle Klassen geladen sind, ist Java dann genauso schnell wie andere Programmiersprachen.

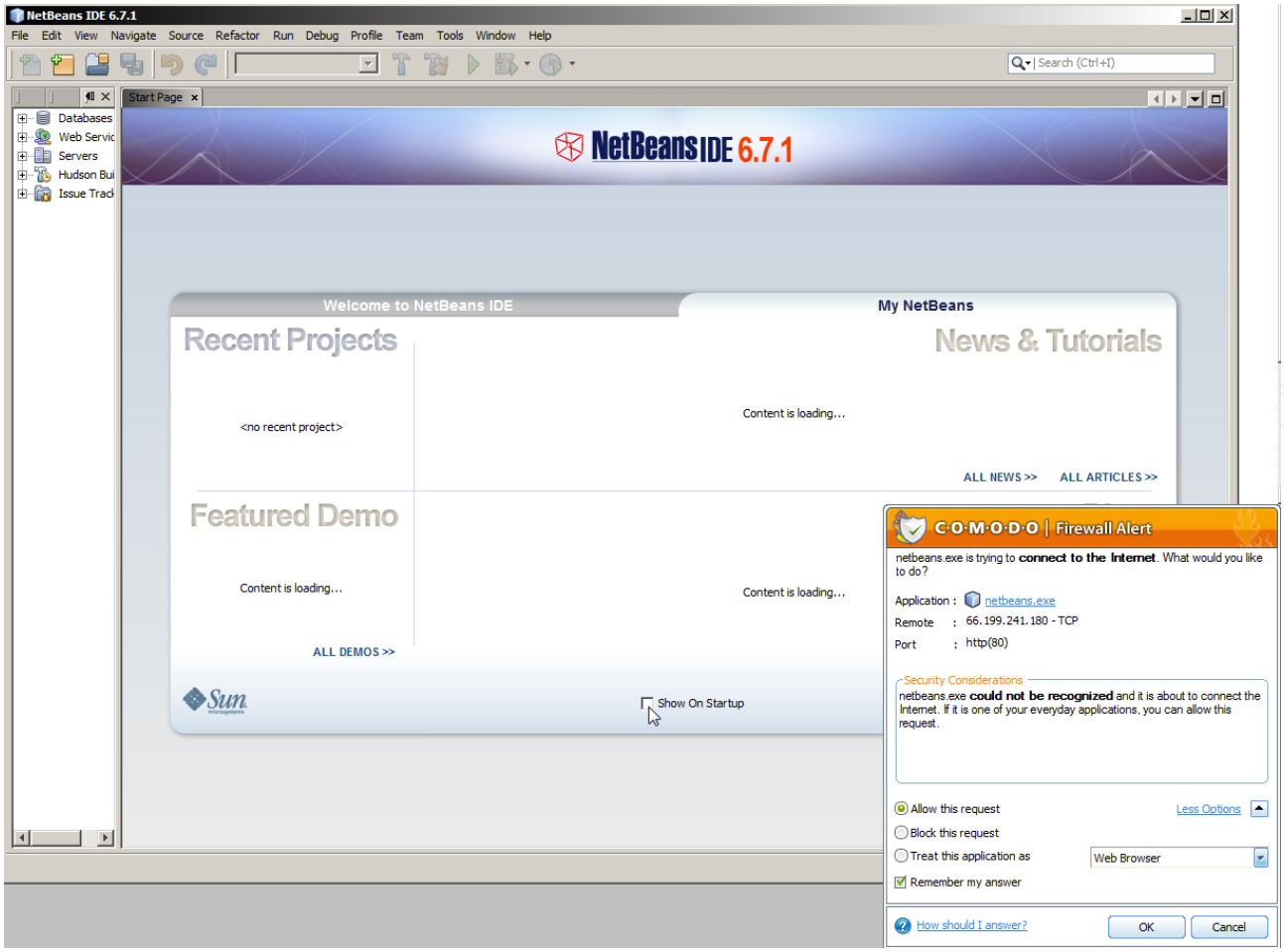




NetBeans lädt am Anfang standardmäßig mehr oder minder interessante Informationen aus dem Internet. Möchte man dies vermeiden, muss der Haken oben-rechts bei „Show On Startup“ weggenommen werden. Mit dem kleinen x bei Start Page wird diese Seite geschlossen.



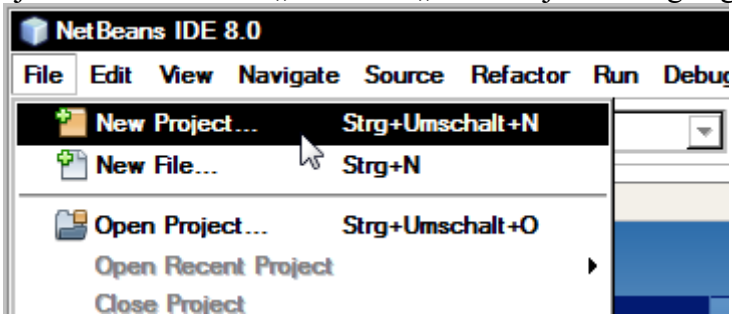
Generell ist zu beachten, dass NetBeans einige Prozesse, u. a. anderem jetzt oder später einen Datenbank- und einen Web-Server startet, so dass man ggfls. seine Firewall- und Virenscanner-Einstellungen anpassen muss.



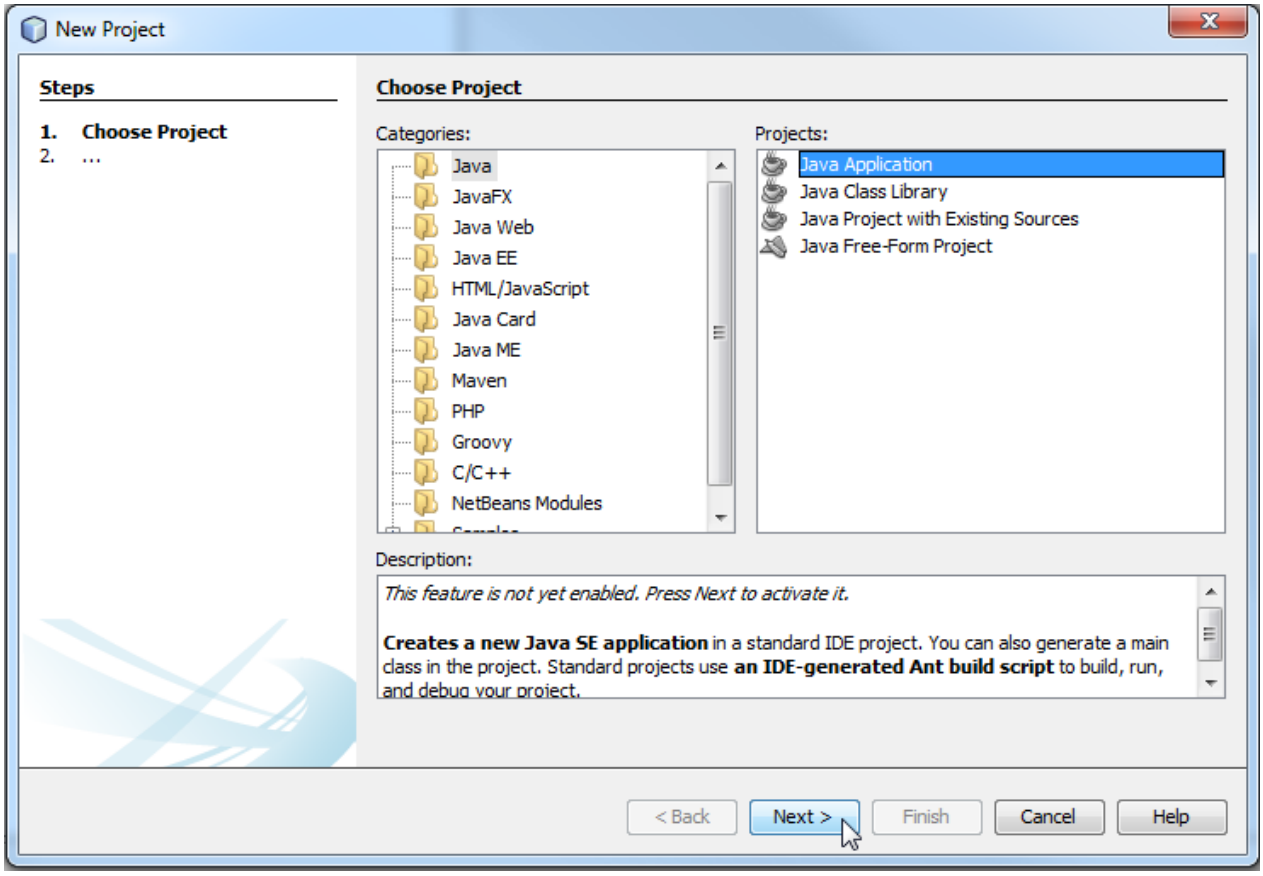
Nach diesem etwas nervigen Einstieg verhält sich NetBeans wieder recht „friedlich“.

3.2 Anlegen eines Java-Projekts

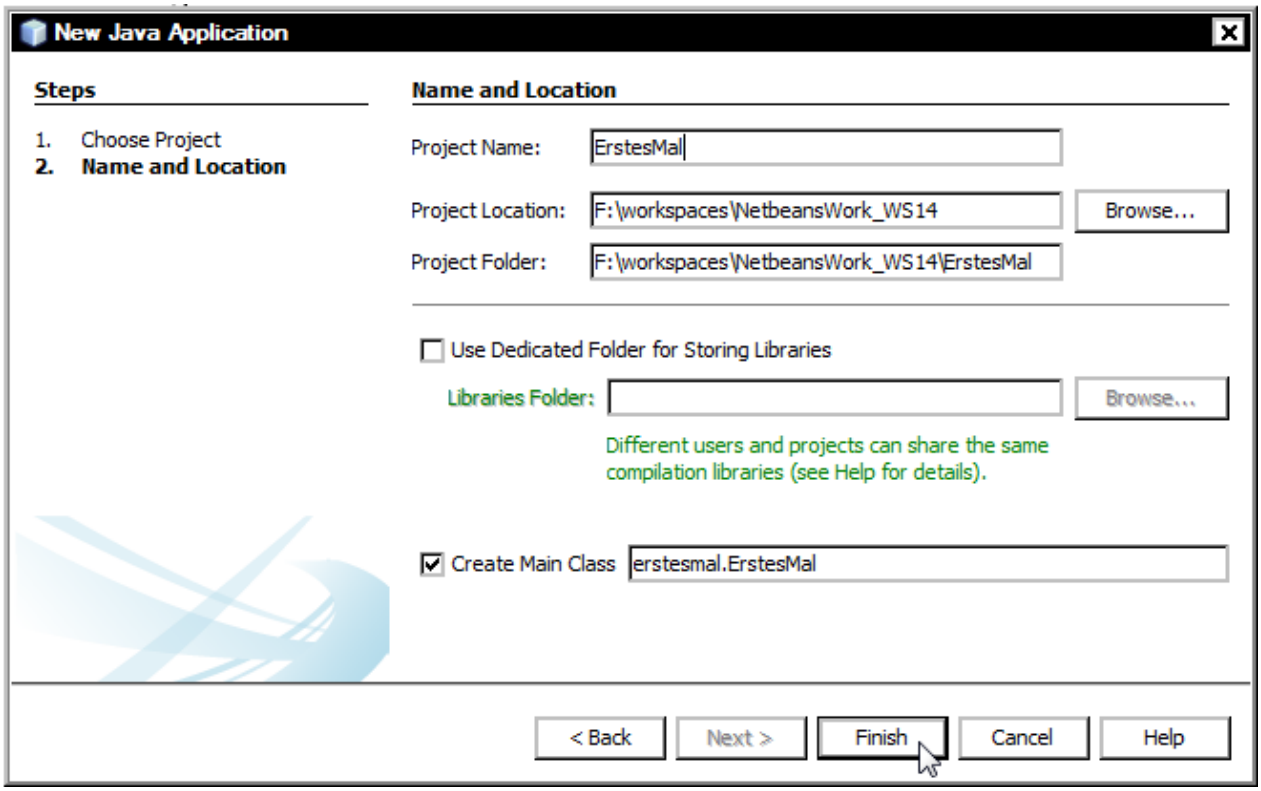
Ein einfaches Java-Projekt kann z. B. über „File“ und „New Project...“ angelegt werden.



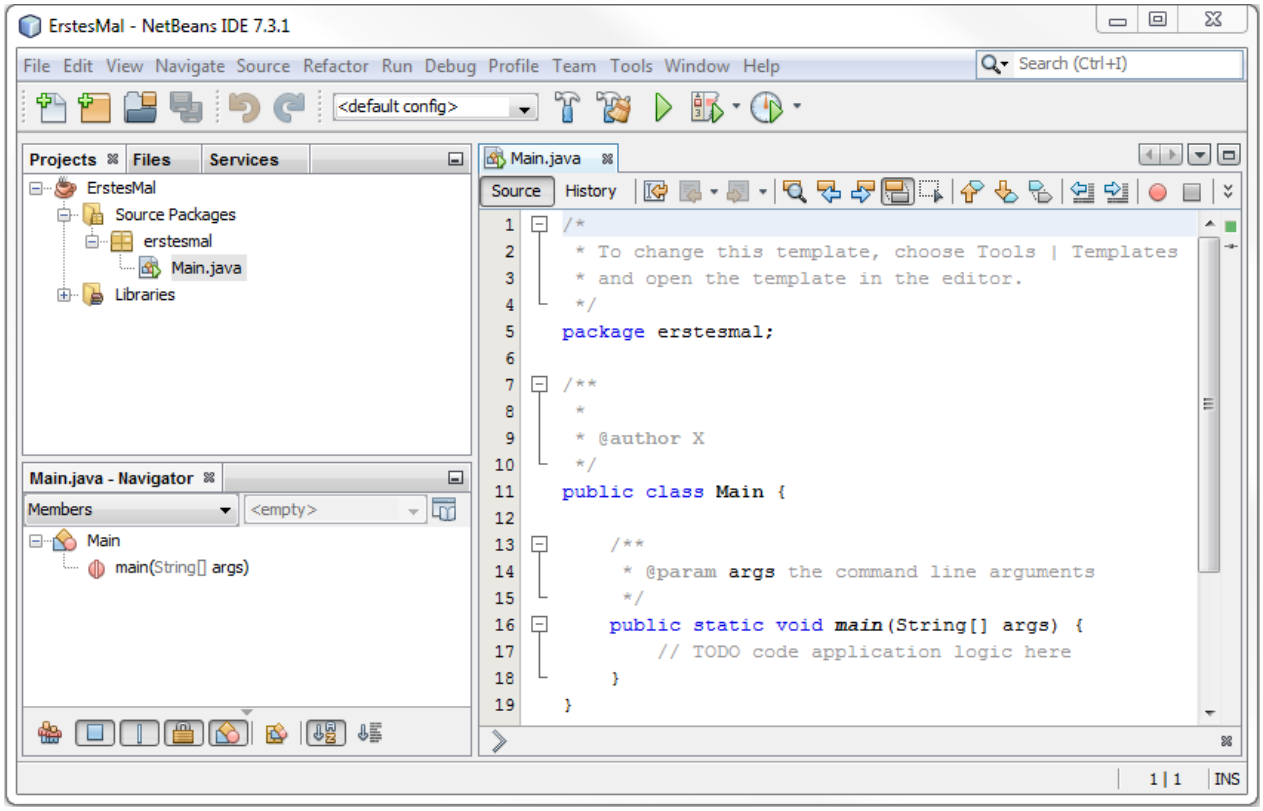
Danach wählt man als „Category“ Java und als „Project“ Java Application und drückt „Next>“.



Nur beim ersten Start ist kurz eine Meldung „Activating Java SE“ zu sehen.
Bei den folgenden Einstellungen muss ein sinnvoller Projektname vergeben werden. Weiterhin ist eine Project Location zu wählen, d. h. ein Verzeichnis, in dem der Projektordner (Project Folder) mit allen Dateien angelegt werden soll. Man kann hier eine Location eingeben, unter der alle späteren Projekte verwaltet werden sollen. Man beachte, dass anders als Eclipse kein zentraler Workspace vorgeschrieben ist. Generell ist es sinnvoll, den Ordner vorher anzulegen, alternativ kann dies auch durch NetBeans passieren, indem er bei der „Project Location“ eingetragen wird.
Ob eine Klasse Main, hier im Paket erstesmal, erzeugt werden soll, ist reine Geschmackssache und kann geändert werden. Generell ist es eine gute Idee, den Startpunkt eines Java-Programms in eine Main-Klasse zu legen.



Danach wird das Projekt angelegt. Im folgenden Beispiel wurde die Main-Class erzeugt; diese wird dann sofort im Editor geöffnet. Die Kommentare sind in NetBeans konfigurierbar.



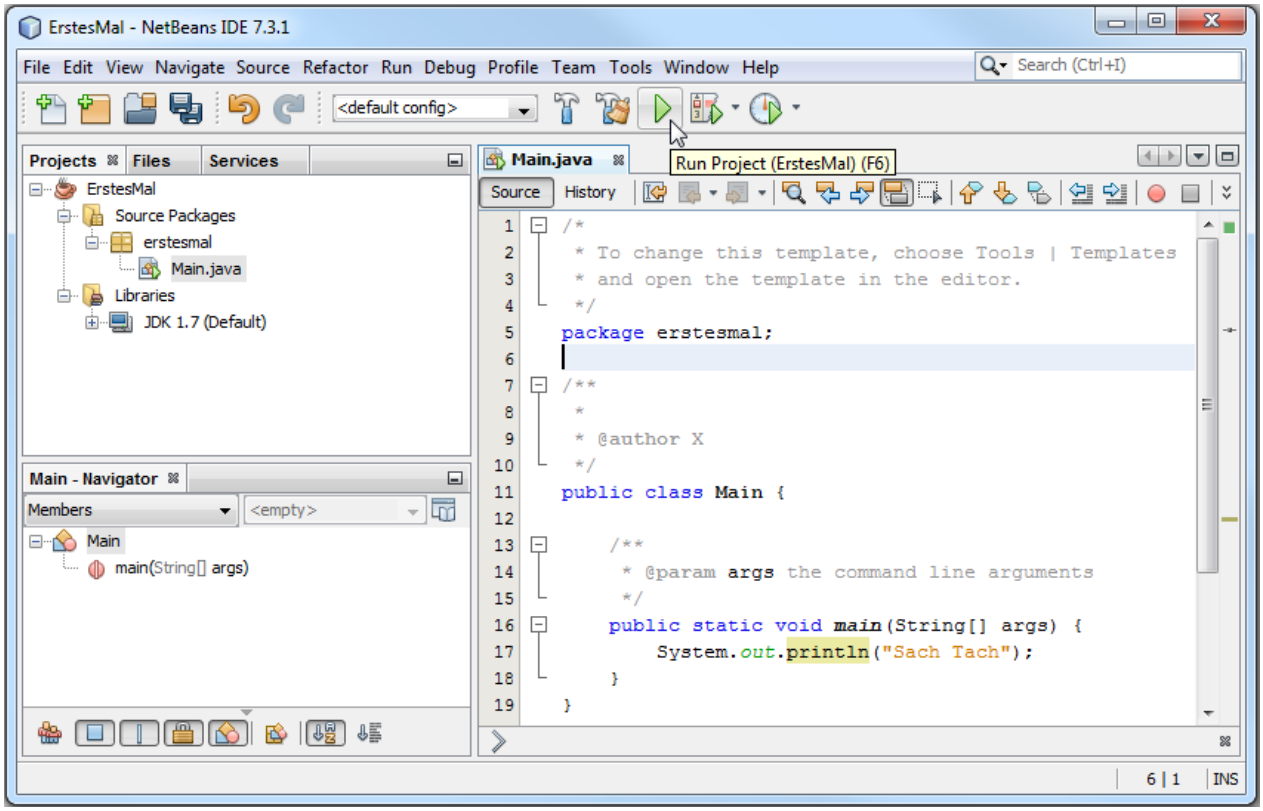
3.3 Ausführen eines Programms

Nachdem ein Programm eingegeben wurde, kann man es z. B. über den grünen Pfeil in der Menü-Leiste starten. Die fette Schrift im Reiter des Editor-Fensters von Main.java zeigt, dass die Datei noch nicht gespeichert wurde, was durch „Run“ automatisch passiert.

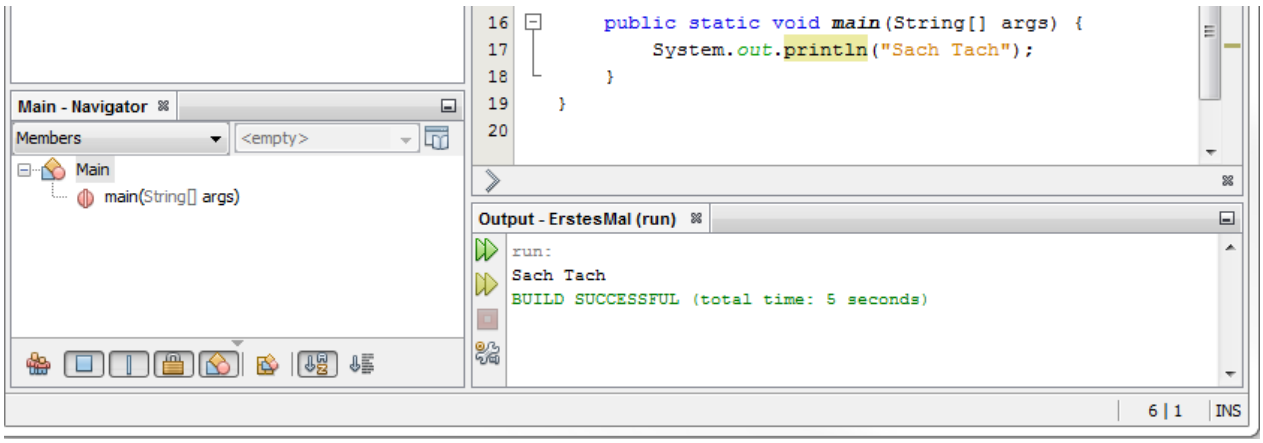
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package erstesmal;

/**
 * @author X
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Sach Tach");
    }
}
```

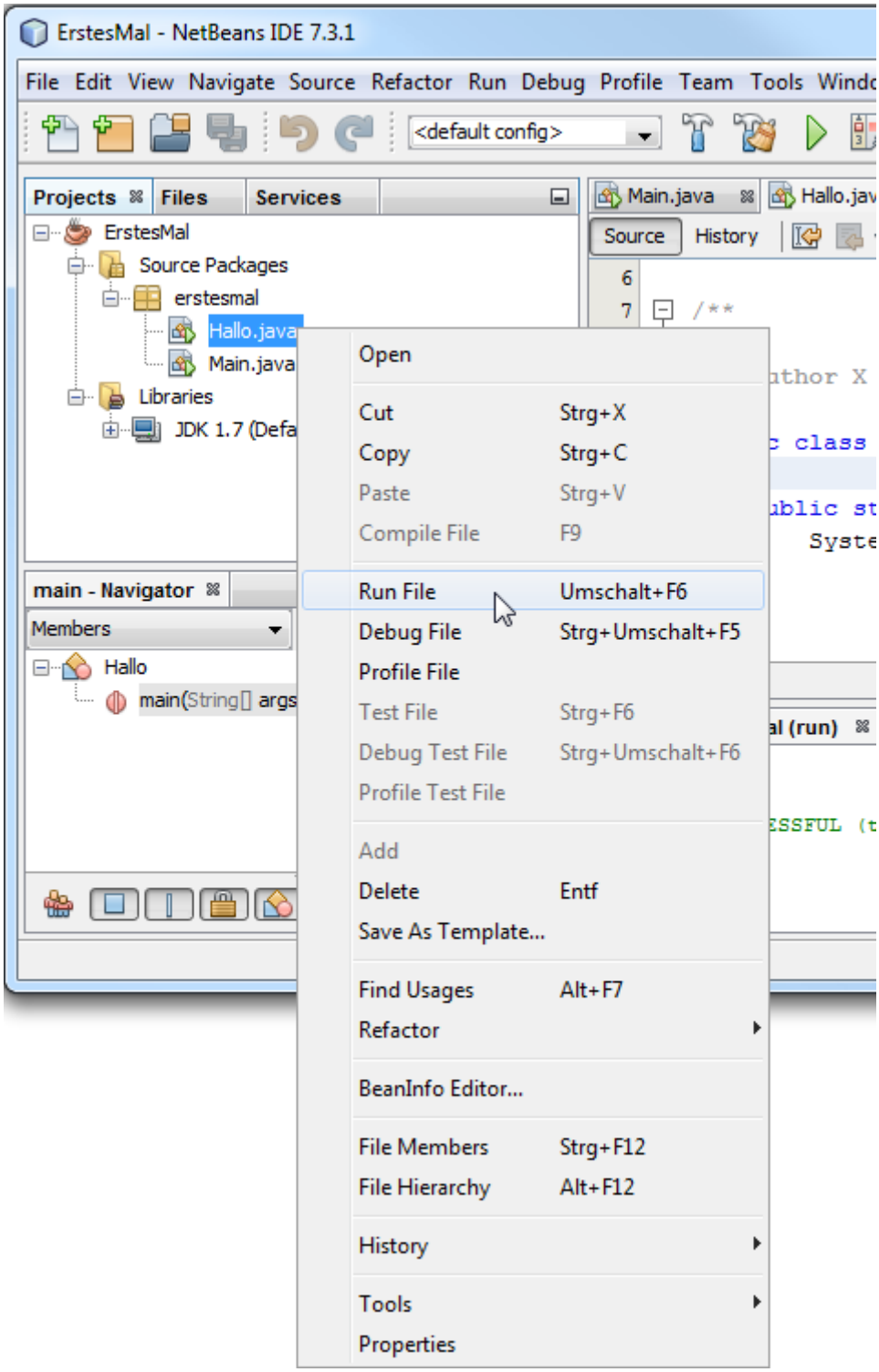


Ausgaben werden unten im Output-Fenster zusammen mit einigen Ausführungsinformationen ausgegeben. Diese Konsole wird auch bei Eingaben benutzt.

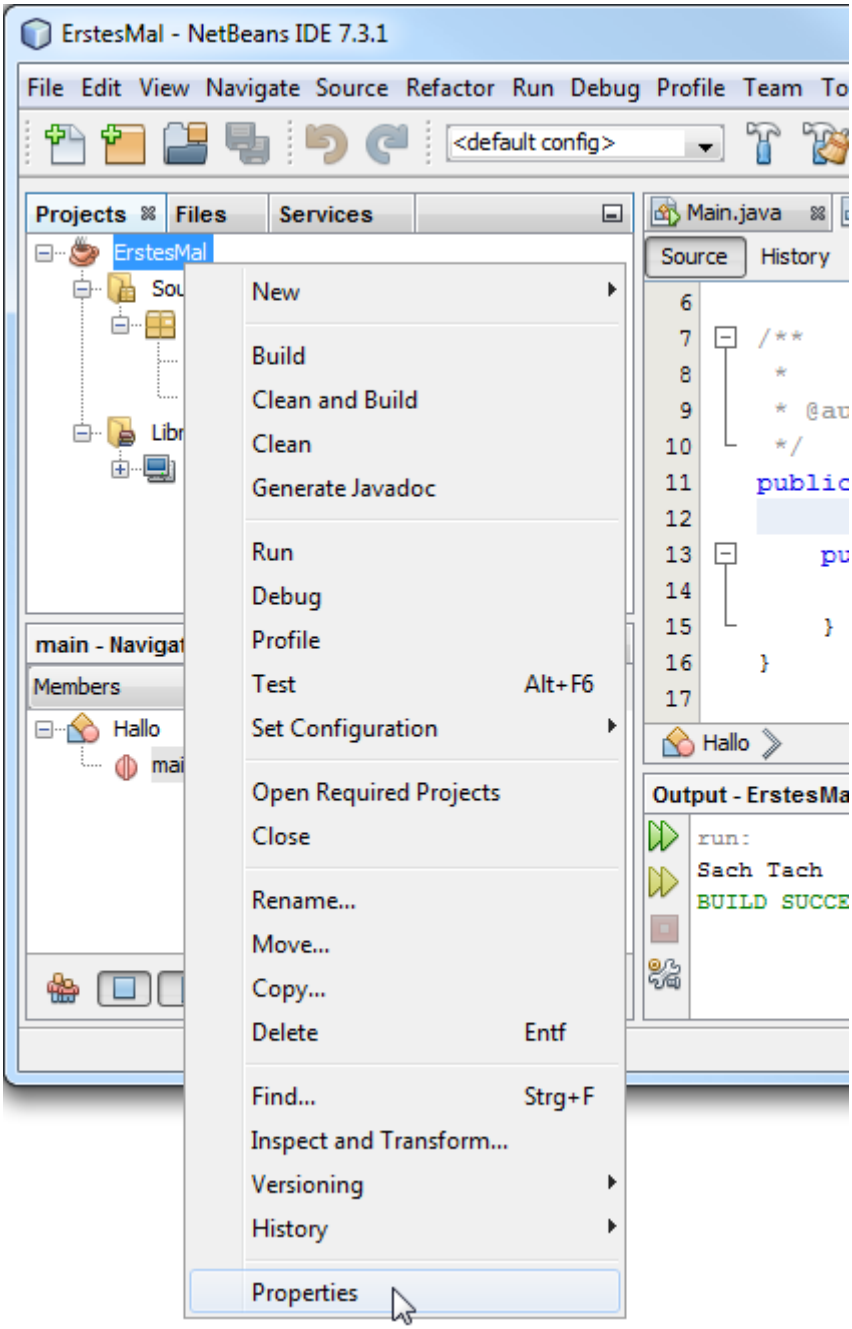


3.4 Mehrere ausführbare Programme

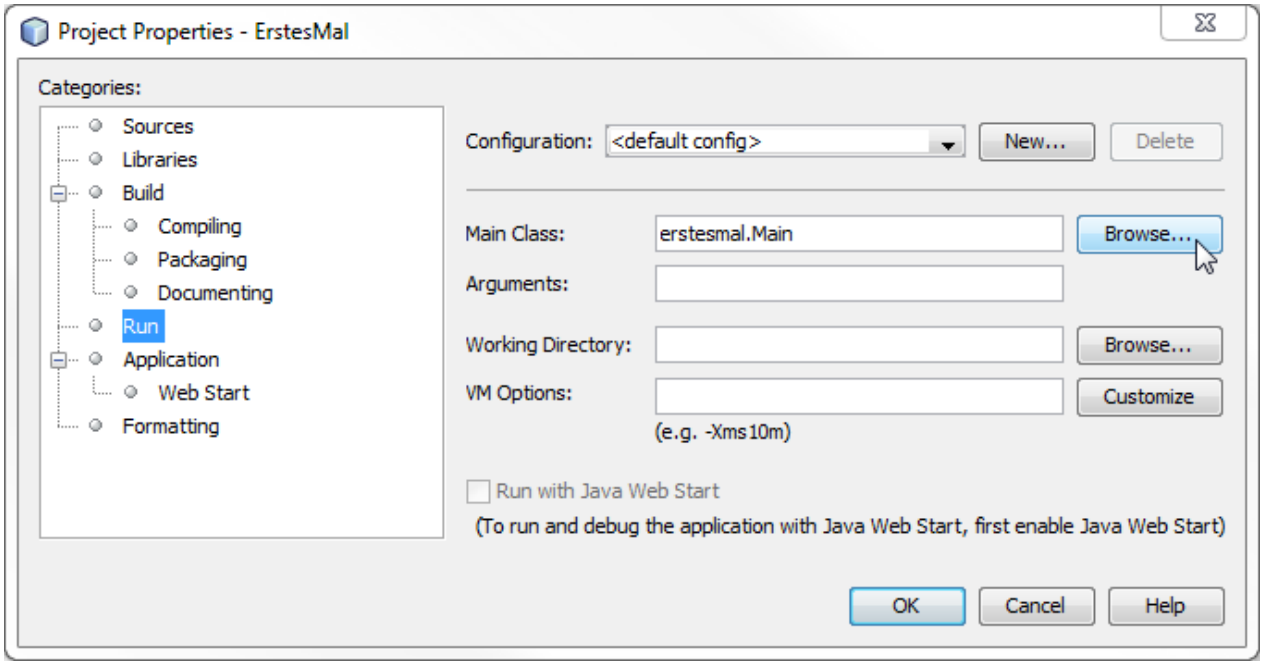
Existieren in einem Projekt mehrere Klassen mit einer startbaren main-Methode, so kann die Klasse durch einen Rechtsklick und der Auswahl „Run File“ direkt gestartet werden. Das folgende Bild zeigt weiterhin, dass ausführbare Klassen mit einem kleinen Dreieck gekennzeichnet sind.



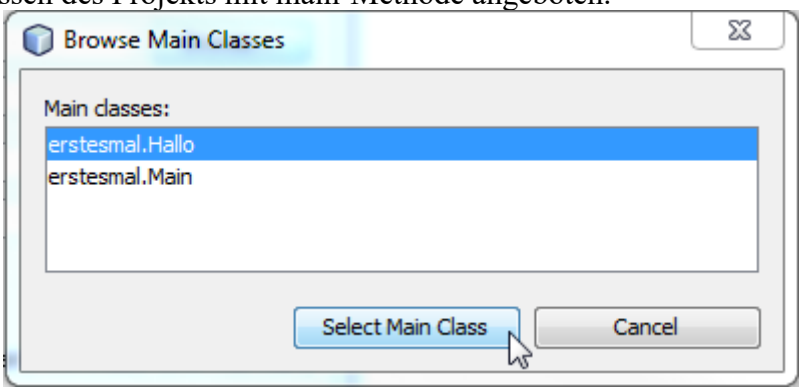
Möchte man, dass eine andere Klasse standardmäßig über den grünen Pfeil in der Werkzeugleiste gestartet wird, muss man die Eigenschaften des Projekts ändern. Dazu wird ein Rechtsklick auf das Projekt gemacht und unten „Properties“ ausgewählt.



Unter „Run“ auf der rechten Seite kann man dann unter „Main Class“ mit „Browse...“ eine neue Klasse auswählen.

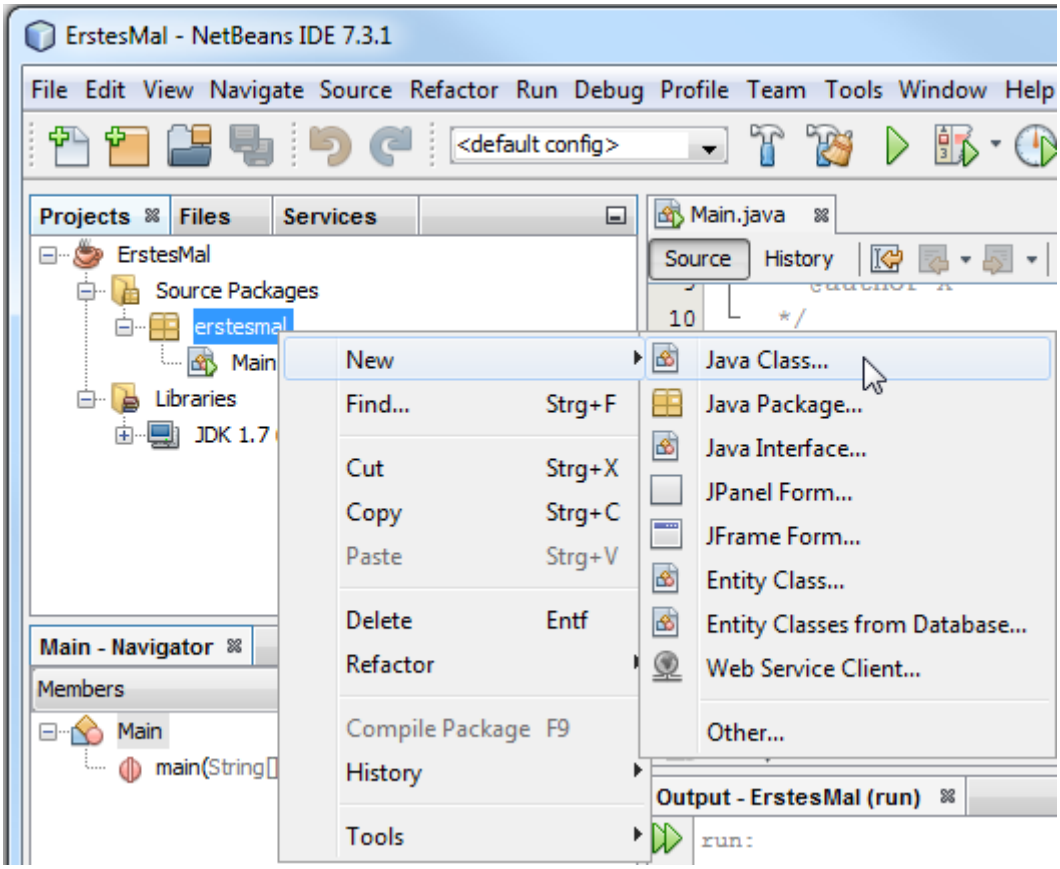


Es werden nur Klassen des Projekts mit main-Methode angeboten.

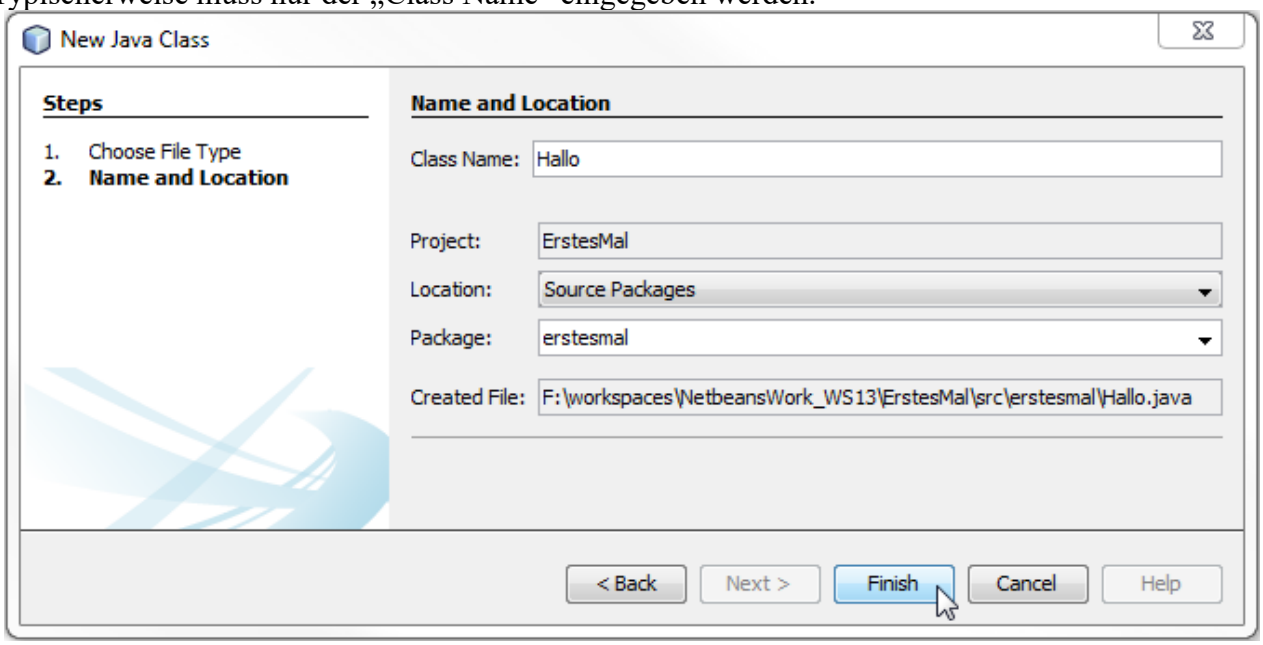


3.5 Anlegen einer neuen Klasse

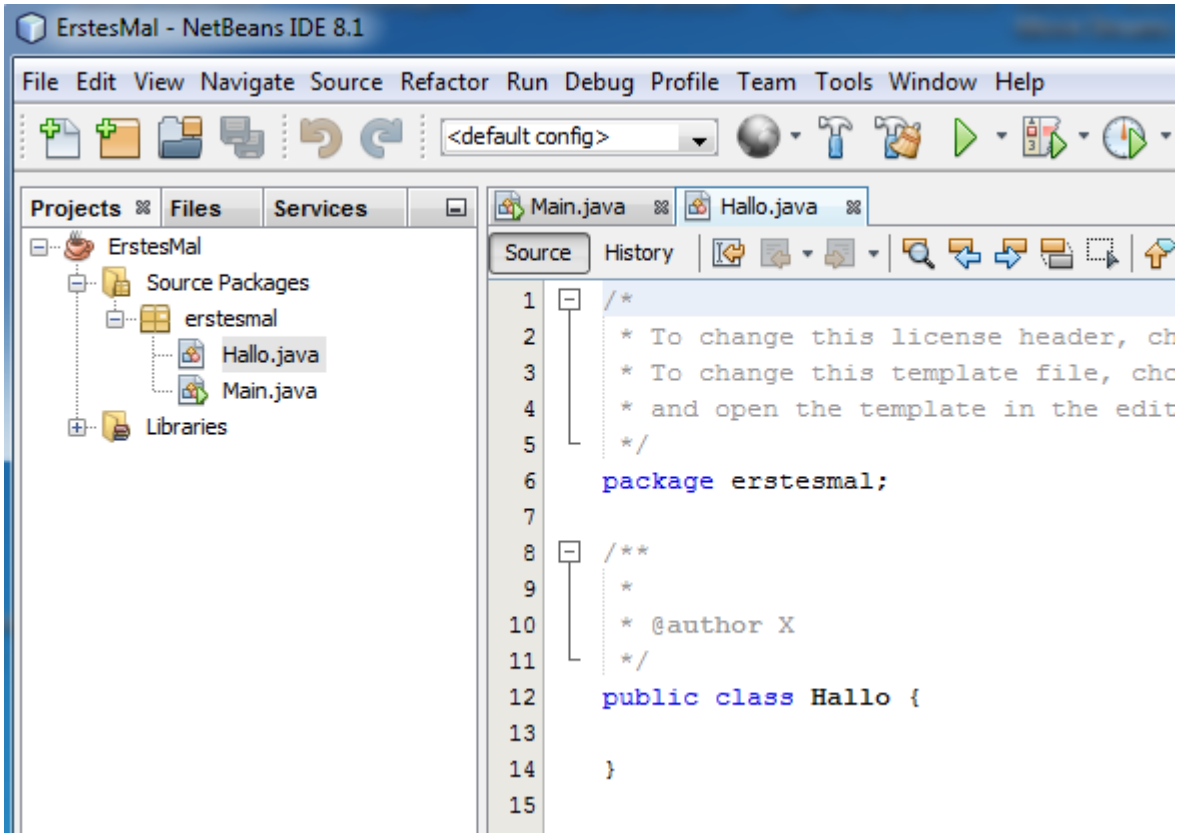
Möchte man eine neue Klasse anlegen, macht man z. B. einen Rechtsklick auf das Paket, in dem die Klasse erstellt werden soll. Man wählt dann „New“ und „Java Class...“.



Typischerweise muss nur der „Class Name“ eingegeben werden.

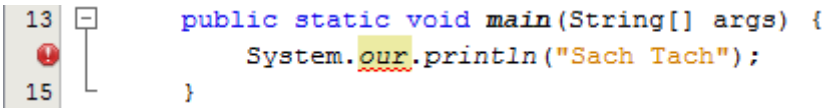


Es wird eine leere Klasse angezeigt, die dann programmiert werden kann.

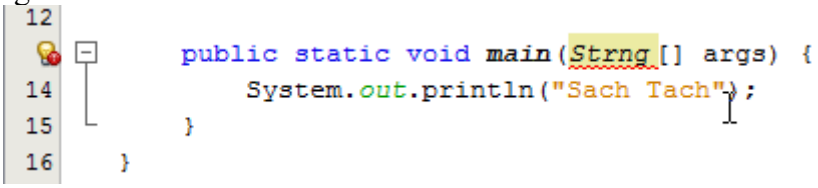


3.6 Syntaxprüfung und Fehlerbearbeitung

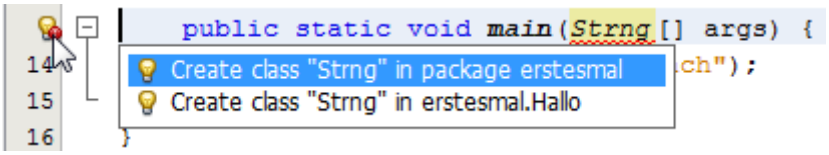
Der Programmtext wird normal eingegeben. Insofern keine Einstellungen verändert wurden, findet direkt eine Syntaxüberprüfung statt. Fehler werden, wie im folgenden Bild gezeigt, mit einem Ausrufungszeichen markiert.



Falls NetBeans Korrekturvorschläge generieren kann, wird dies durch eine Glühbirne zusammen mit dem Fehlersymbol gekennzeichnet.

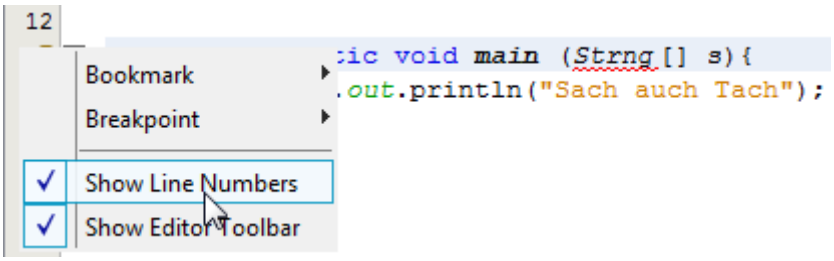


Die Korrekturvorschläge sind durch einen Linksklick auf das Symbol zugänglich, müssen aber nicht immer sinnvoll sein. Die Korrekturvorschläge können dann mit einem Mausklick ausgewählt werden.



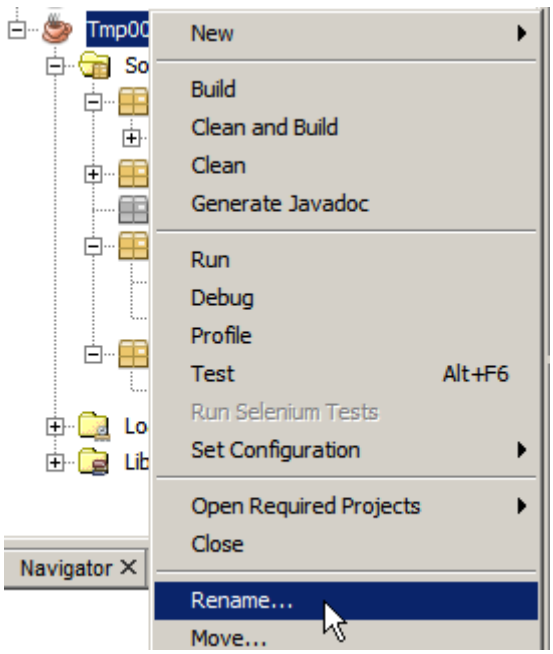
3.7 Zeilennummern

Generell hilfreich ist die Anzeige von Zeilennummern, die man z. B. direkt im Editor aus- und einschalten kann, dazu wird auf die kleine Spalte links neben dem Programmcode mit der rechten Maustaste geklickt und „Show Line Numbers“ ausgewählt. Die Einstellung gilt dann für alle Editor-Fenster.

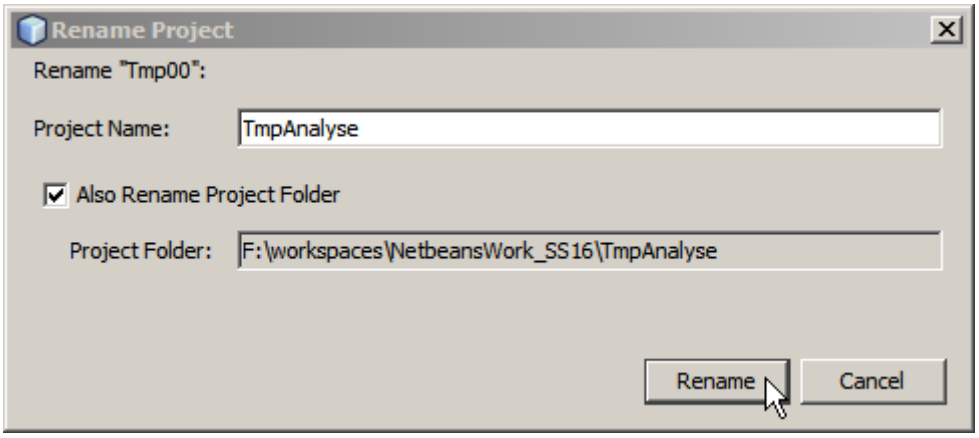


3.8 Umbenennung von Projekten

Zur Umbenennung eines Projekts wird ein Rechtsklick auf dem Projekt gemacht und „Rename“ gewählt.

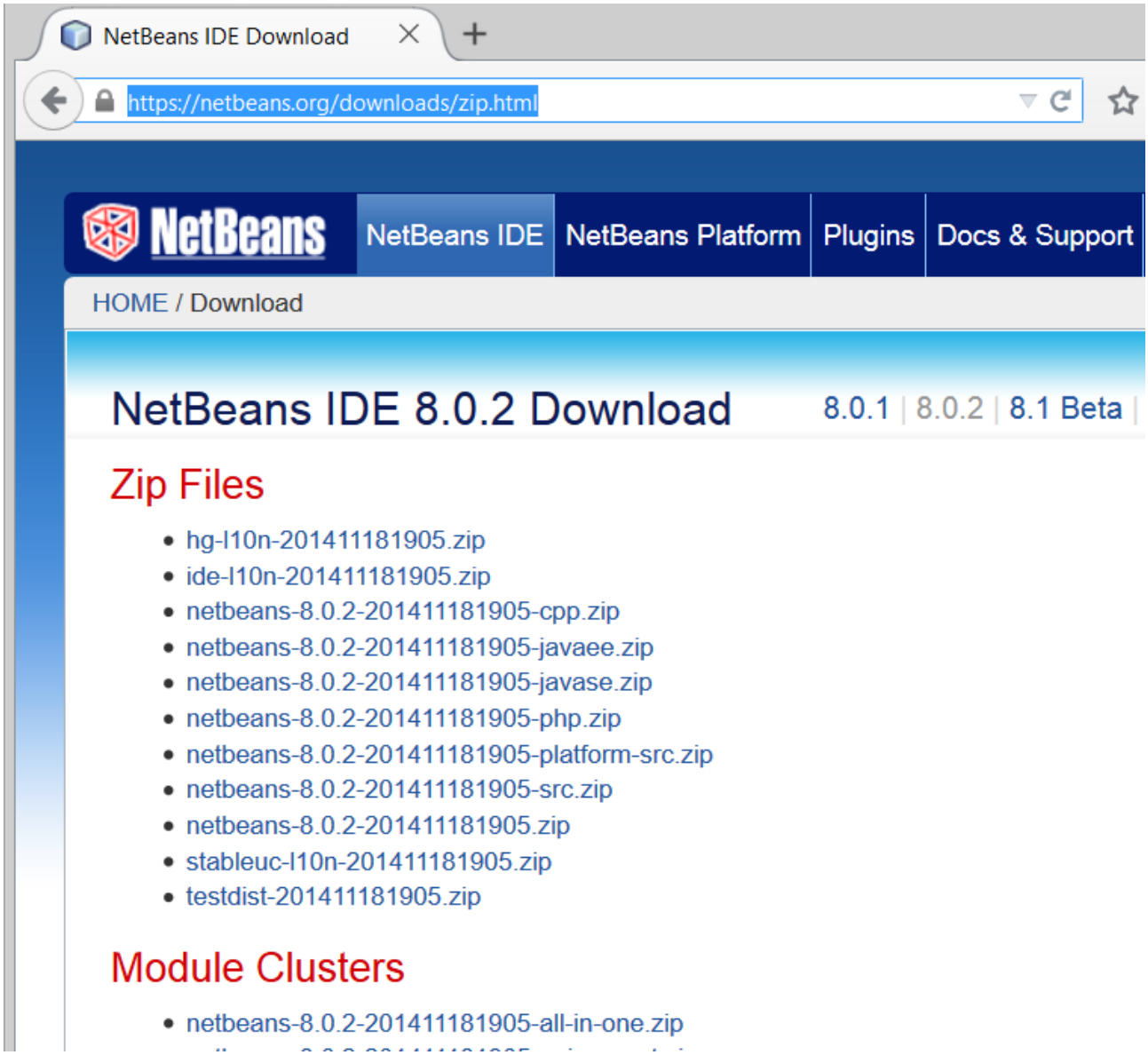


Nun kann oben der neue Projektname eingegeben werden. Weiterhin ist es sehr sinnvoll, den Haken bei „Also Rename Project Folder“ zu setzen, da so das Verzeichnis mit den Projektinhalten den gleichen Namen wie das Projekt erhält.



3.9 Portables NetBeans

Portable Software ist dadurch charakterisiert, dass sie nicht installiert werden muss und so direkt, meist ohne Administrationsrechte, nutzbar ist. Dies hat den wesentlichen Vorteil, dass man unabhängig von der auf einem Rechner installierten Software arbeiten kann, wenn z. B. die Programme auf einem USB-Stick vorliegen. Dieser Ansatz ist weit verbreitet, einen guten Einblick in die Möglichkeiten liefert http://portableapps.com/. Oft reicht USB 2.0 aus, um die Programme zu nutzen. Als Variante kann man die portablen Programme auf einen Rechner kopieren und dort ausführen. Zu beachten ist, dass viele Programme nicht portabel sind, da sie Einträge in der Registry benötigen oder von festen Pfaden abhängig sind.
NetBeans selbst nutzt intensiv das Verzeichnis des aktuellen Nutzers, um Konfigurationsdateien abzulegen. Dieser Ort ist änderbar, so dass man NetBeans portabel nutzen kann. Als Grundlage wird dabei nicht die NetBeans-Installation sondern eine von NetBeans zur Verfügung gestellte zip-Datei, die unter https://netbeans.org/downloads/zip.html erhältlich ist, genutzt. Von einer auf Sourceforge gehosteten Variante wird hier abgeraten, da sie nur Java SE unterstützt.

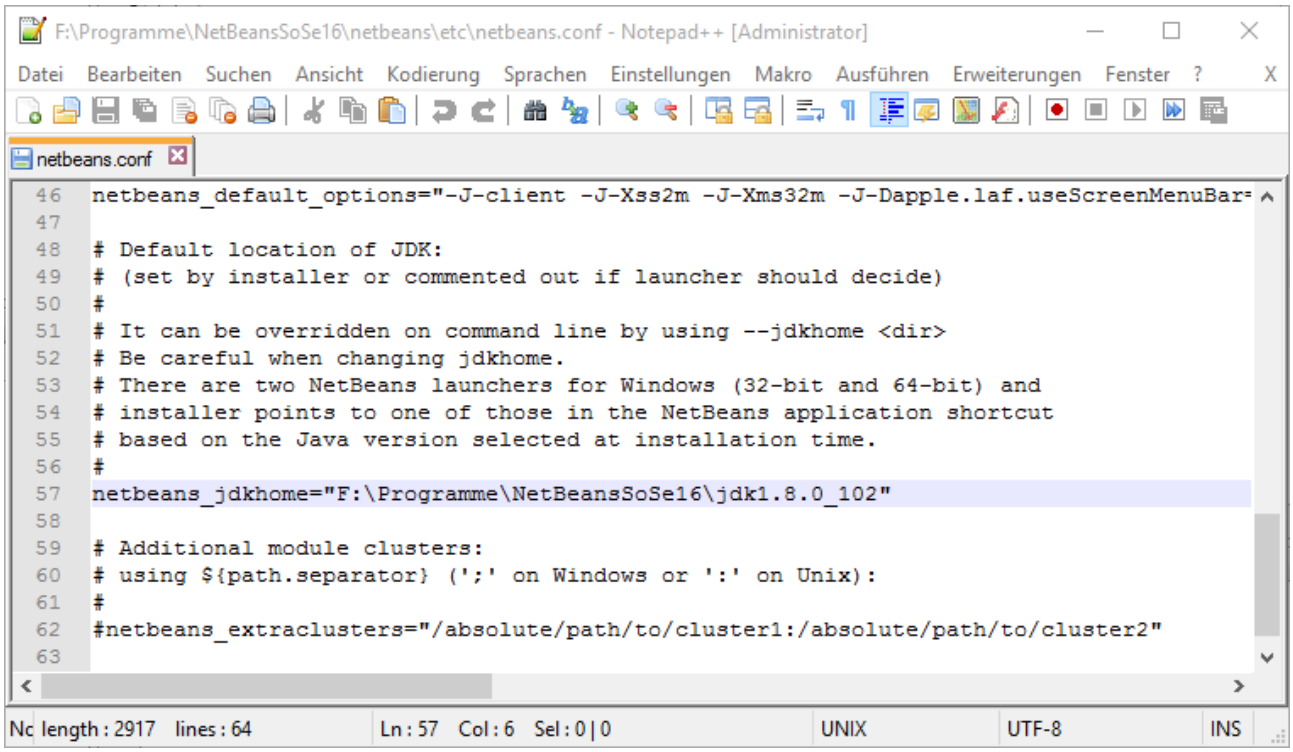


Die zip-Datei der ausgewählten Variante kann an einem beliebigen Ort ausgepackt werden. Das Startprogramm netbeans.exe befindet sich im bin-Verzeichnis. Da der Ordner mit den NetBeans-spezifischen Einstellungen über einen Aufrufparameter eingestellt wird, ist es sinnvoll eine Start-Datei als bat-Datei anzulegen. Ein Beispiel im NetBeans-Ordner kann wie folgt aussehen.

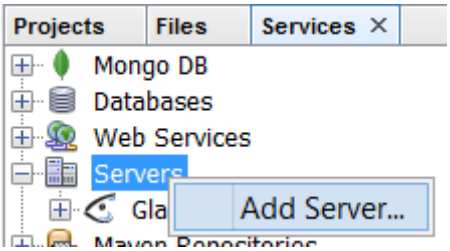
```
set JAVA_HOME=F:\Programme\Java\jdk1.8.0_51
set JDK_HOME=F:\Programme\Java\jdk1.8.0_51
@start "" "%~dp0.\bin\netbeans.exe" --userdir "%~dp0../.userdir" --console suppress
```

Die ersten beiden Zeilen dienen zur Nutzung einer Java-Version, die sich auf einem USB-Stick mit dem Laufwerksbuchstaben F befindet. Diese Einstellungen sind nicht notwendig, wenn eine Installation auf dem genutzten Rechner vorliegt, zeigt aber die Nutzungsmöglichkeiten von angepassten Systemvariablen. Zur Java-Nutzung werden einfach die Installationsordner auf den USB-Stick kopiert. In der eigentlichen Startzeile wird mit %~dp0 auf das aktuelle Verzeichnis

zugegriffen, in dessen Unterverzeichnis die auszuführende Datei netbeans.exe liegt. Über den Parameter --userdir wird das Verzeichnis angegeben, in das Nutzereinstellungen kopiert werden, dies ist im Beispiel der Ordner .userdir, der als Unterordner parallel zum NetBeans-Ordner angelegt wird. Der Ordner darf sich nicht im NetBeans-Installationsordner befinden. Der letzte Parameter sorgt dafür, dass kein Konsolenfenster während der Nutzung offen ist, kann also weggelassen werden. Diese Informationen sind im Wesentlichen http://www.netbeans-forum.de/viewtopic.php?t=5328 entnommen.
Sollte sich auf dem genutzten Rechner keine Java-Installation befinden, reichen obige Einstellungen leider nicht aus und der Pfad zur Java-Installation muss zusätzlich in die Datei netbeans.conf im etc-Unterordner des NetBeans-Ordners eingetragen werden. Dazu wird das Kommentar-Zeichen # in der zu netbeans_jdkhome gehörenden Zeile gelöscht und der Pfad in Anführungsstrichen eingetragen, was hier in Zeile 57 der Fall ist.



Nach dem Start kann NetBeans normal eingerichtet werden. Beim Anlegen von Datenbanken kann z. B. ein Ordner auf dem USB-Stick genutzt werden, wie in „6.2 Anlegen einer neuen Datenbank“ gezeigt. Möchte man die Java-Version ändern, kann man „4.7 Installation einer neueren/weiteren Java-Version“ folgen.
Im Zip-Verzeichnis befindet sich kein GlassFish-Server, dieser ist aber ebenfalls portabel und kann in einem beliebigen Verzeichnis, in dem Programme ausgeführt werden können, liegen. Der Server wird dann über „New Server“ angelegt, was auch in „2.2 Installation eines eigenen GlassFish-Servers“ genauer beschrieben ist.



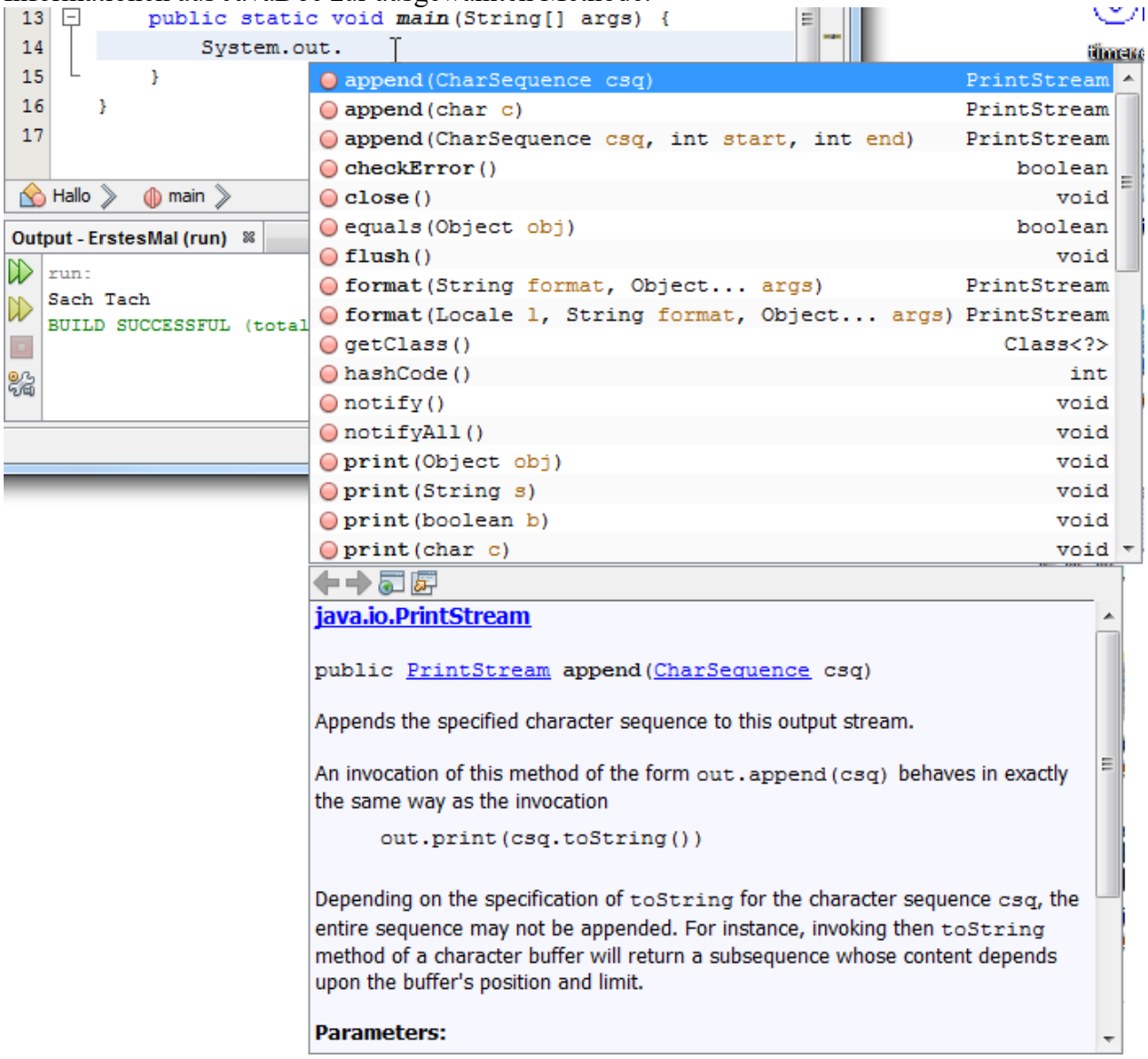
Auch der Scene Builder für JavaFX ist portabel und kann, wie in „11.4 Installation des Scene Builders“ gezeigt, eingebunden werden.
Die Installation von Plugins, wie in „15.1 Installation eines NetBeans-Plugins am Beispiel NBMongo“ gezeigt, verläuft in allen NetBeans-Varianten identisch.

4 Hilfreiche NetBeans-Funktionalität im Java-Alltag

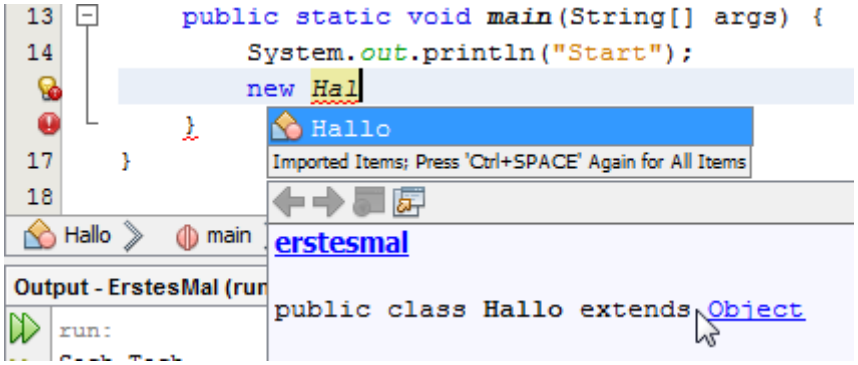
Hier werden nur wenige grundlegende Tipps gegeben. Für Freunde von Tastaturkürzeln, z. B. um System.out.println() mit fünf Tasten (sout<Tab-Taste>) einzugeben, sei auf die Hilfe (Help) von NetBeans verwiesen.

4.1 Code-Vervollständigung

Im Editor-Fenster werden beim Tippen automatisch Vorschläge zur Code-Vervollständigung generiert, die einfach mit der Maus-Taste gewählt werden können. Weiterhin erhält man Informationen aus JavaDoc zur ausgewählten Methode.

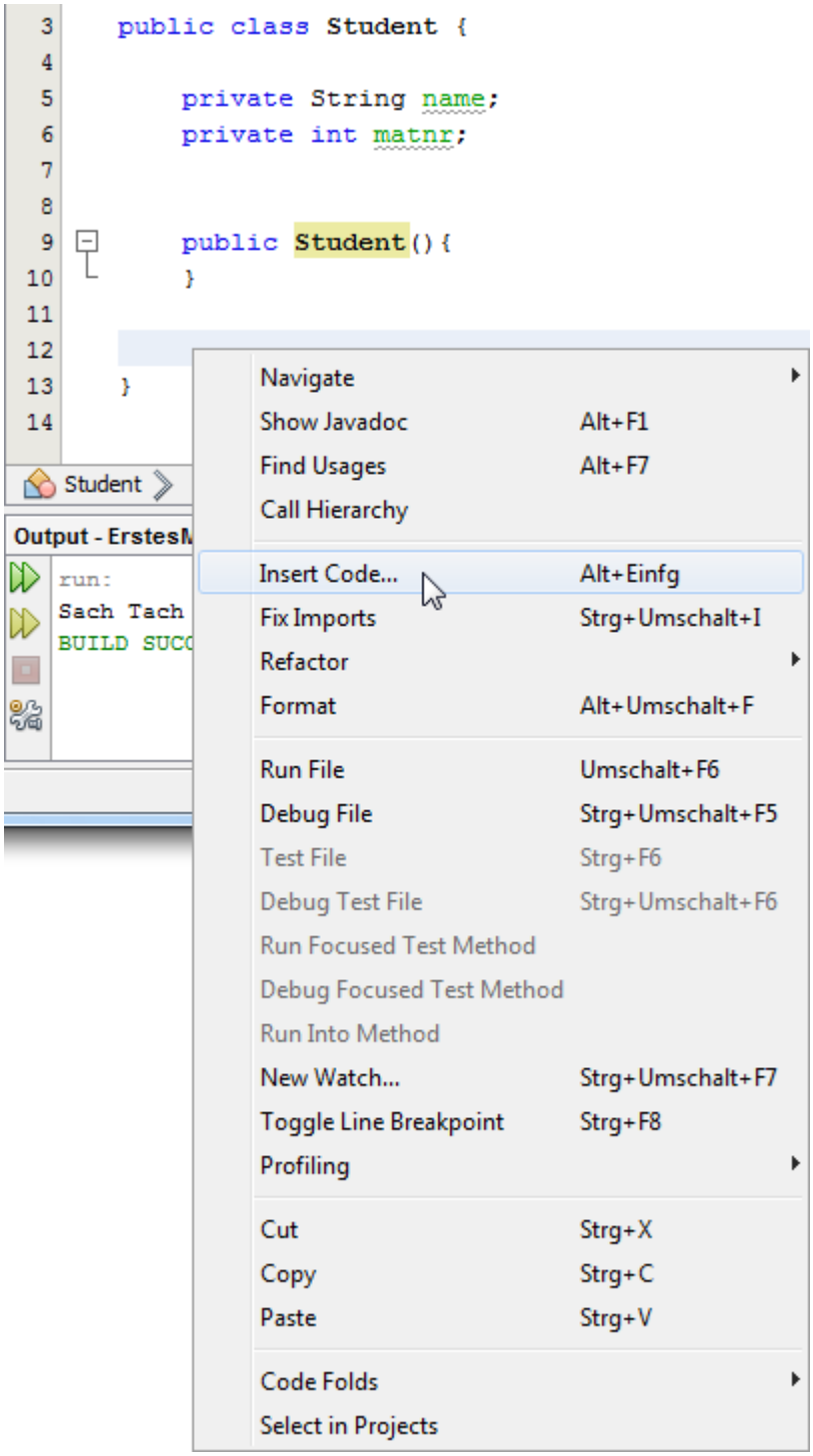


Möchte man die Code-Vervollständigung selbst aktivieren, drückt man „Strg“+„Leertaste“ zusammen. Das folgende Beispiel zeigt, dass es auch aus Entwicklersicht sinnvoll ist, eigene Klassen zu kommentieren, da Kommentare hier angezeigt würden.

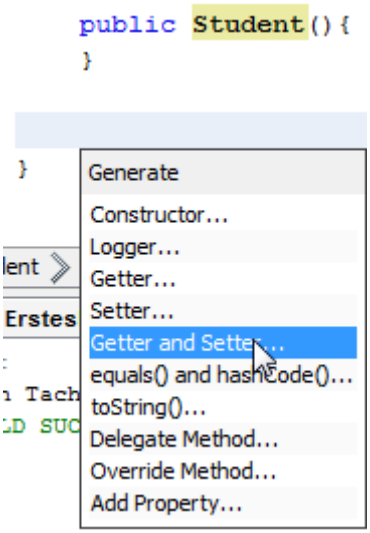


4.2 Generierung von Gettern und Settern

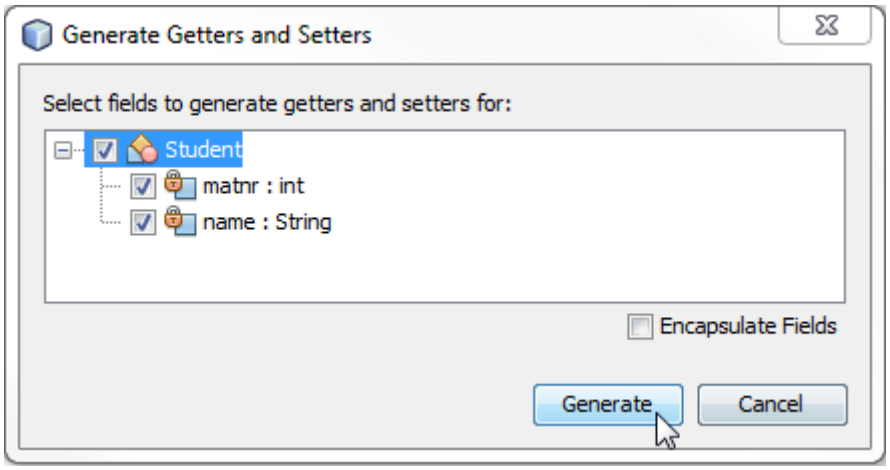
Häufig schreibt man zunächst die Exemplarvariablen (Attribute) einer Klasse auf und muss dann die passenden get- und set-Methoden schreiben. Diese können auch direkt von NetBeans erzeugt werden. Ausgangspunkt soll folgende Klasse sein, bei der man an der Stelle an der die Methoden generiert werden sollen, die rechte Maustaste drückt und dann „Insert Code ...“ wählt.



Man sieht dann, welche Generierungsmöglichkeiten existieren. Hier wird „Getter and Setter ...“ gewählt.

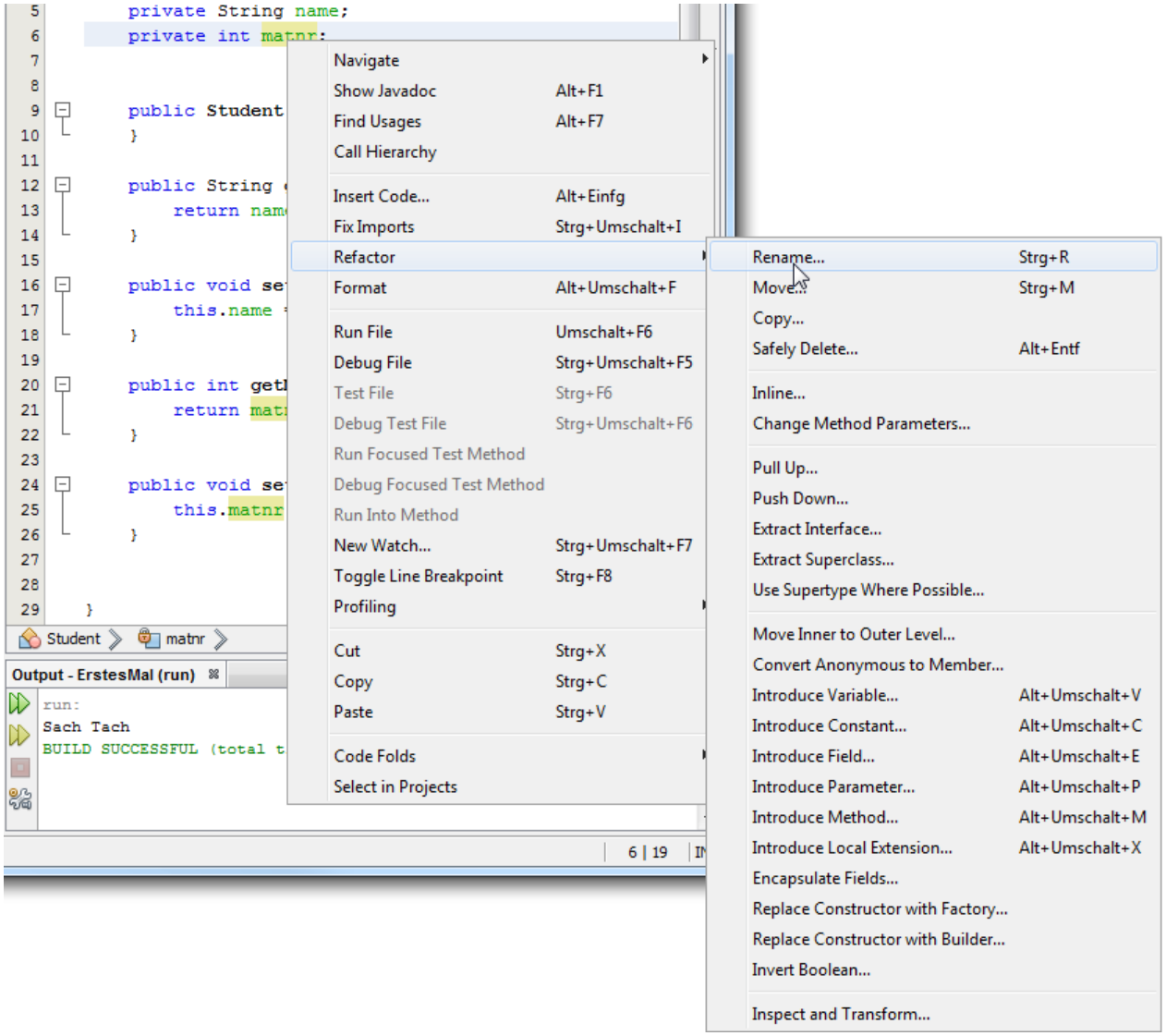


Im folgenden Fenster kann man dann auswählen, für welche Exemplarvariablen die Methoden generiert werden sollen.

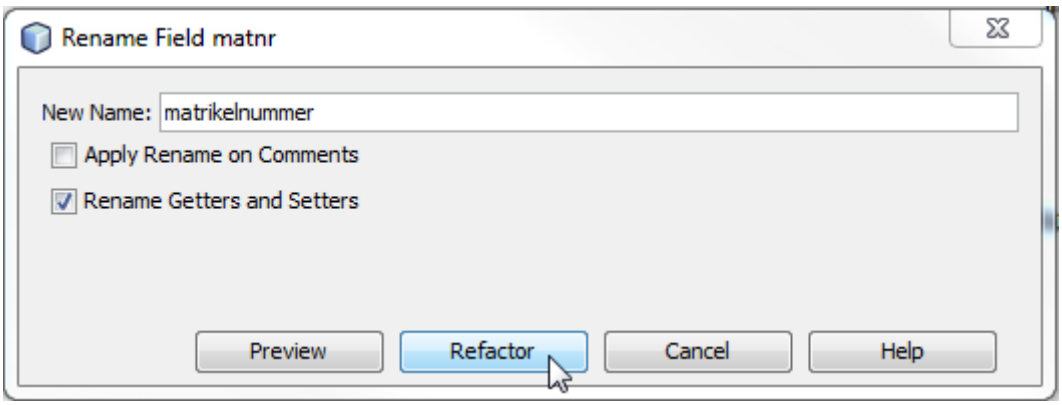


4.3 Umbenennungen / Refactoring

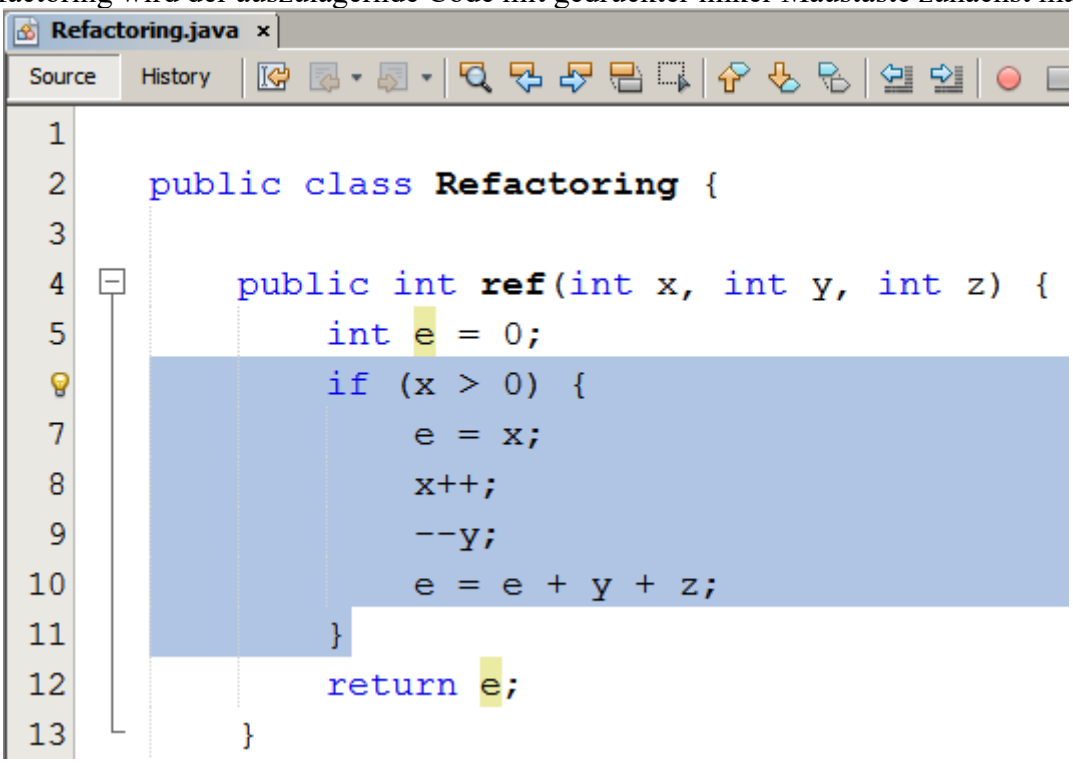
Refactoring behandelt generell den Umbau von Software, so dass Anforderungen nach Wartbarkeit, Performance und Wiederverwendbarkeit besser erfüllt werden. In der Praxis wird eine Refactoring-Phase häufig nach einer längeren Entwicklungsphase eingeschoben, um die Software aufzuräumen. Diese Schritte werden von NetBeans in mehreren Varianten unterstützt, von denen einige wenige hier vorgestellt werden.
Bei Umbenennungen von Variablen oder Methoden ist nur zu beachten, dass es sich um eine Form des Refactoring handelt. Konkret wird das umzubenennende Element, hier die Exemplarvariable matnr mit einem Rechtsklick selektiert und „Refactor“ und dann „Rename...“ ausgewählt.



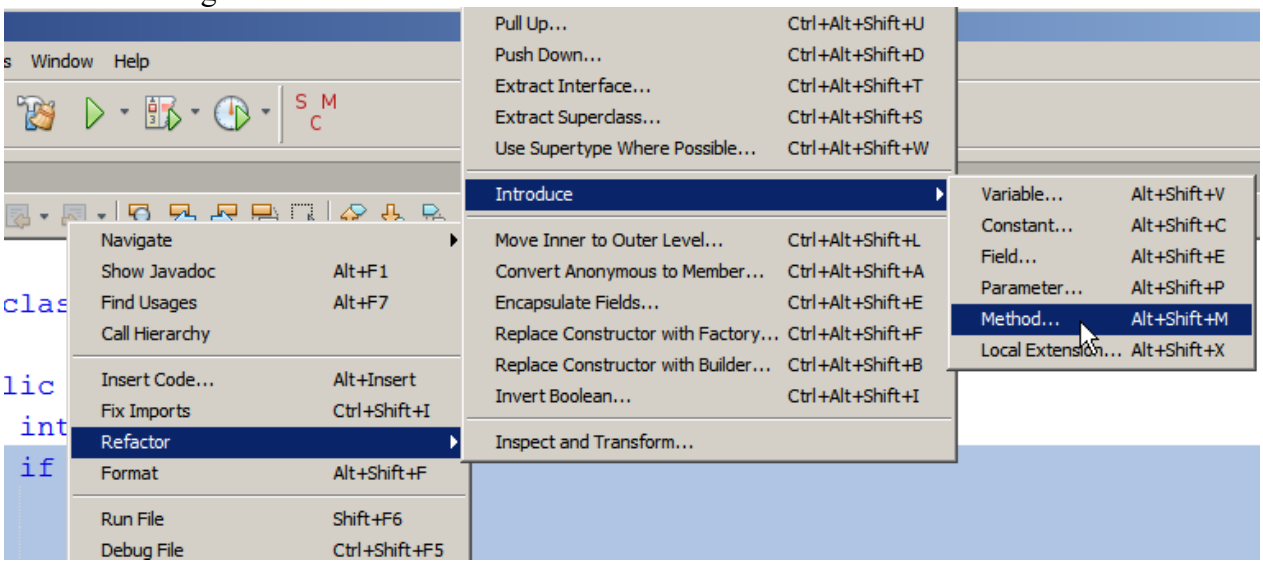
Danach kann der neue Name einfach eingegeben werden. Bei komplexeren Aktion ist die angebotene Vorschau sinnvoll. Man beachte, dass Methodennamen von get- und set-Methoden nicht angepasst werden.



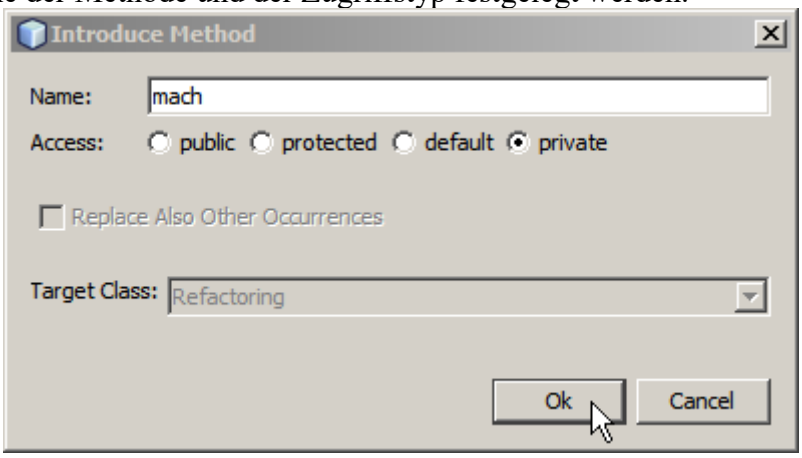
Eine weitreichendere Möglichkeit besteht darin, aus einer gegebenen Methode Teile dieser Methode in eine neue Methode auszulagern. Dies kann die Lesbarkeit von Programmen wesentlich erhöhen, weiterhin können so Code-Verdopplungen vermieden oder korrigiert werden. Man beachte, dass diese Schritte nur automatisierbar sind, wenn die Parameter der Methode und ein eventuell benötigter eindeutiger Rückgabetyp berechnet werden können.
Zum Refactoring wird der auszulagernde Code mit gedrückter linker Maustaste zunächst markiert.



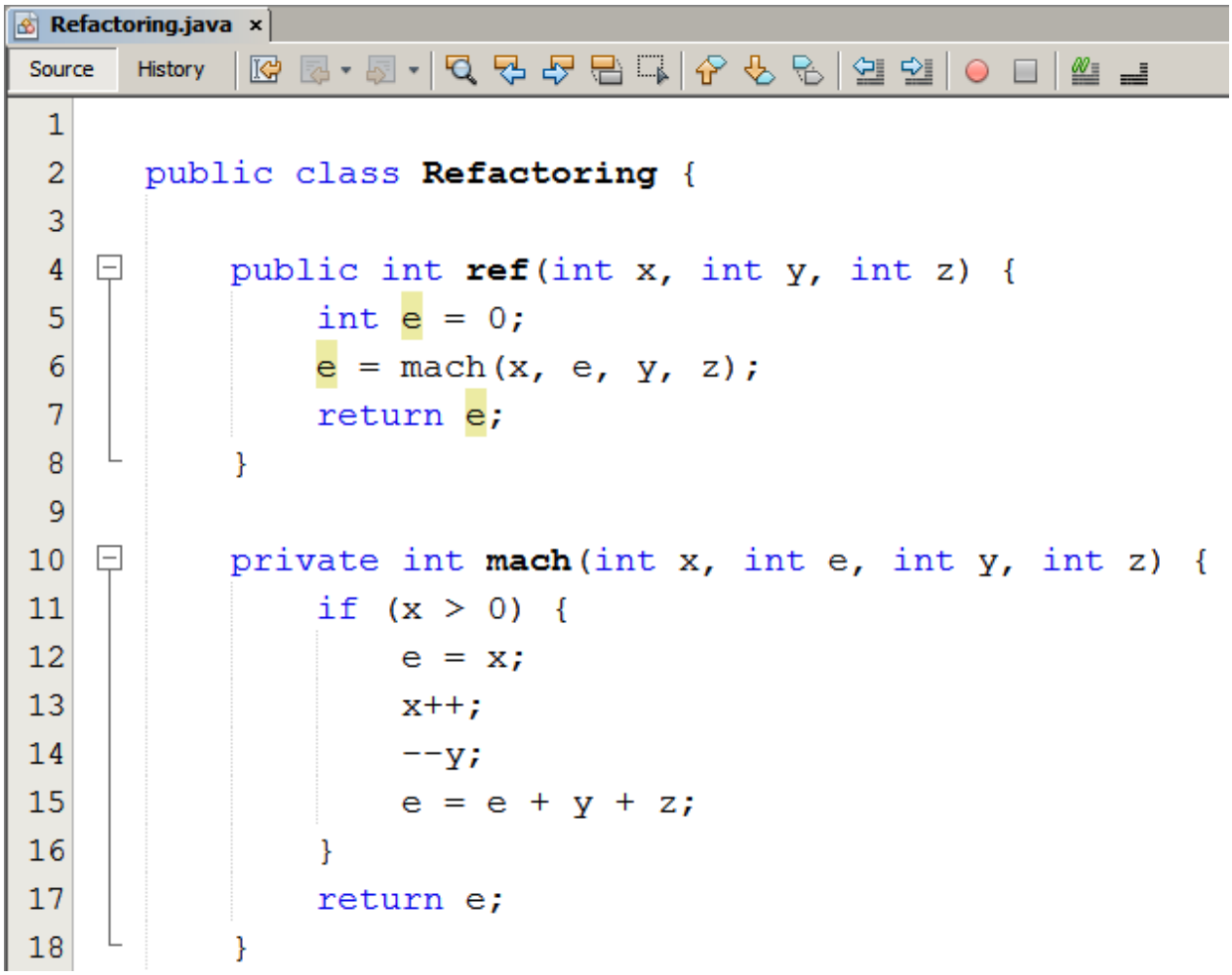
Dann wird auf dem markierten Code ein Rechtsklick gemacht und „Refactor -> Introduce -> Method…“ ausgewählt.



Nun kann der Name der Methode und der Zugriffstyp festgelegt werden.

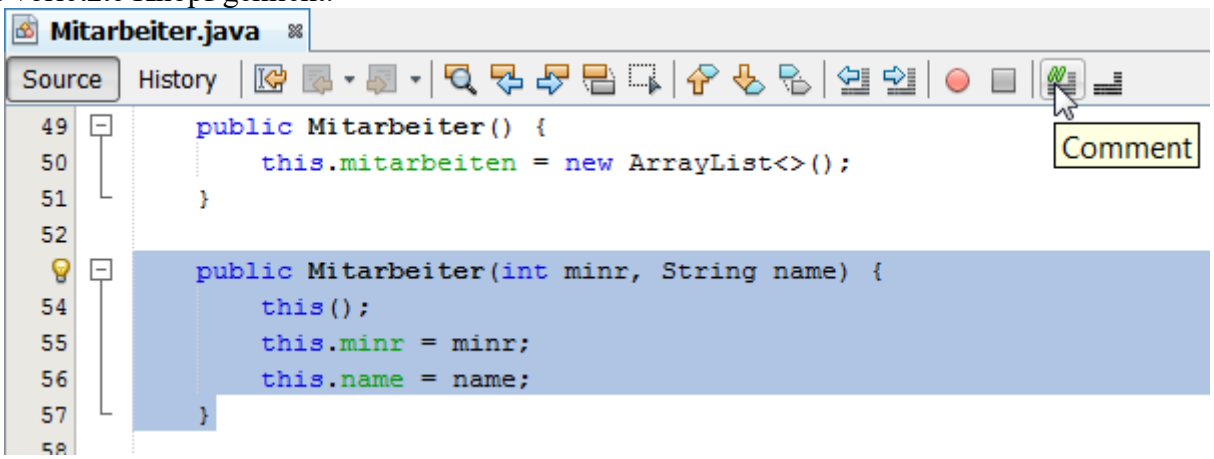


Die neue Methode befindet sich dann im Quellcode und kann bei Bedarf bearbeitet und verschoben werden.

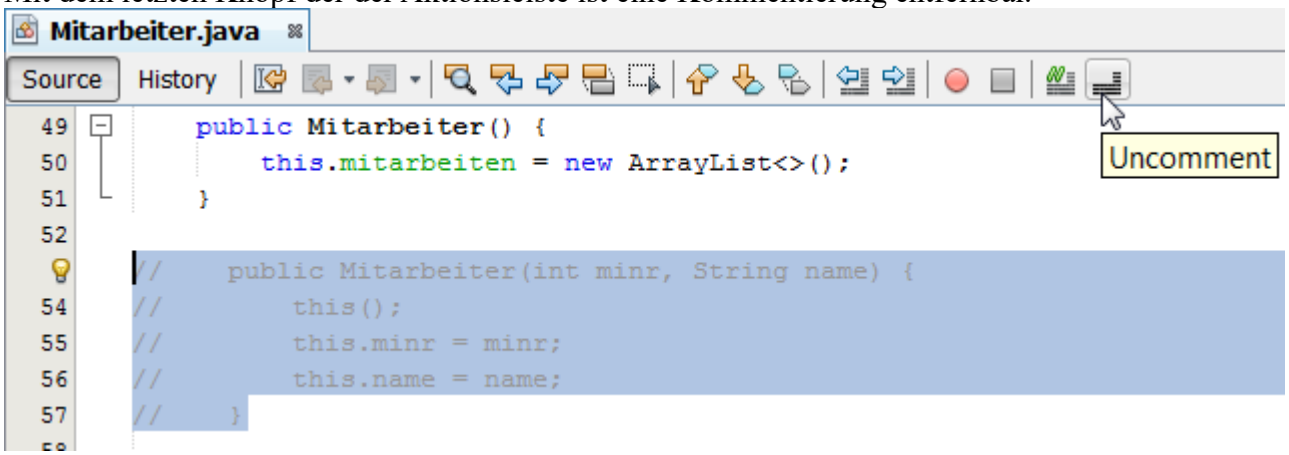


4.4 Kommentierung

NetBeans unterstützt die Möglichkeit, mehrere Code-Zeilen einfach auszukommentieren und später auch wieder die Kommentierung zu entfernen. Um mehrere Zeilen auszukommentieren, werden diese zunächst mit gedrückter linker Maustaste markiert und dann in der Aktionsleiste unterhalb des Reiters der vorletzte Knopf geklickt.



Mit dem letzten Knopf der der Aktionsleiste ist eine Kommentierung entfernbar.



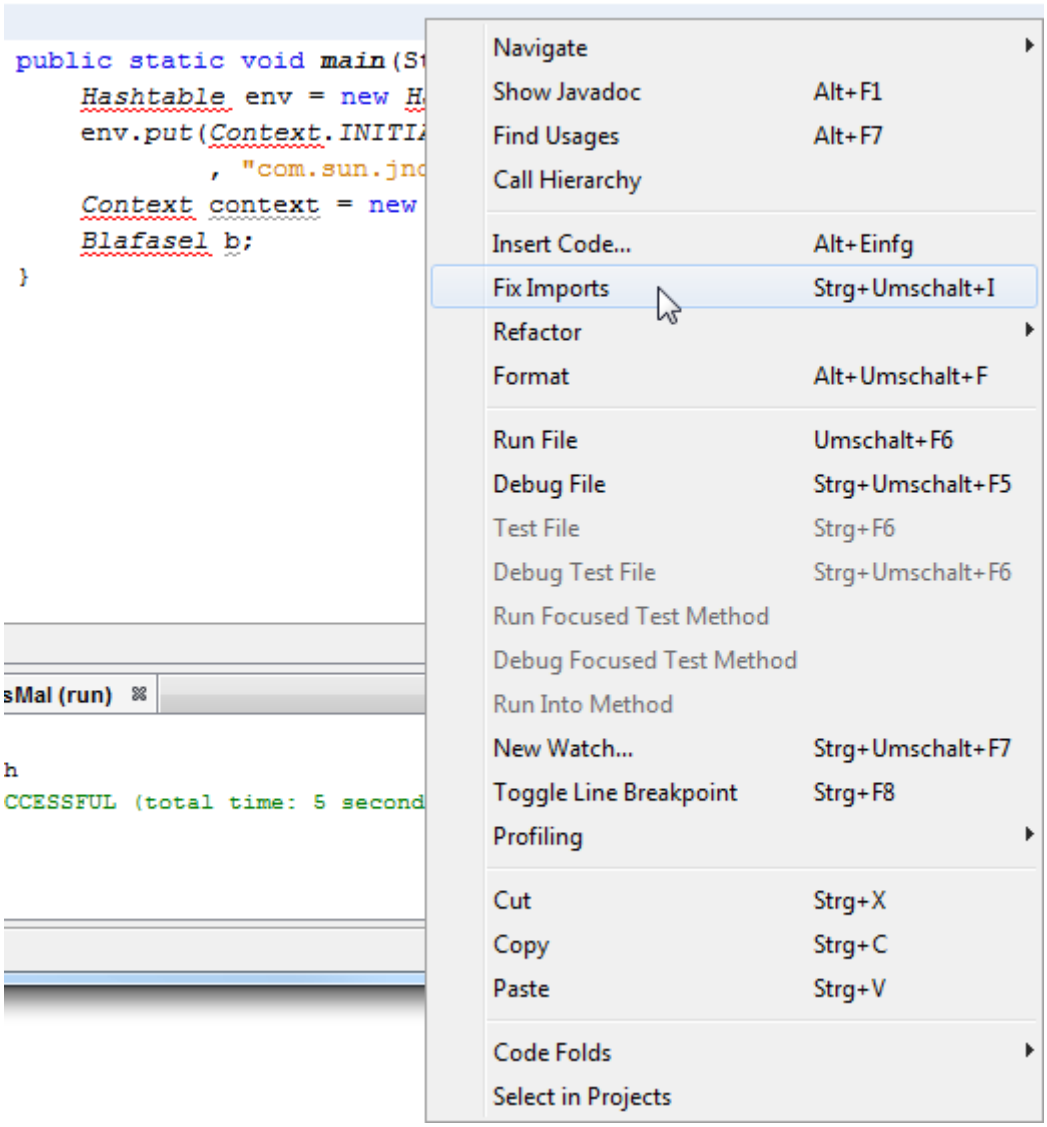
Wird der Kommentierungsknopf doppelt gedrückt wird die Kommentierung der Kommentierung durchgeführt. Dies ist zum Glück mit dem anderen Knopf wieder umkehrbar.

4.5 Organisation von Imports

Häufig liegen genutzte Klassen in Java-Paketen, die nicht automatisch referenziert werden und als import eingebunden werden müssen. Diese import-Beziehungen müssen auch häufig aufgelöst werden, wenn man Programmfragmente aus Anleitungen kopiert. Statt diese von Hand zu lösen, gibt es in NetBeans hierfür eine direkte Unterstützung. Als Beispiel wird folgende Implementierung genutzt, für die noch keine import-Zeilen ergänzt wurden.

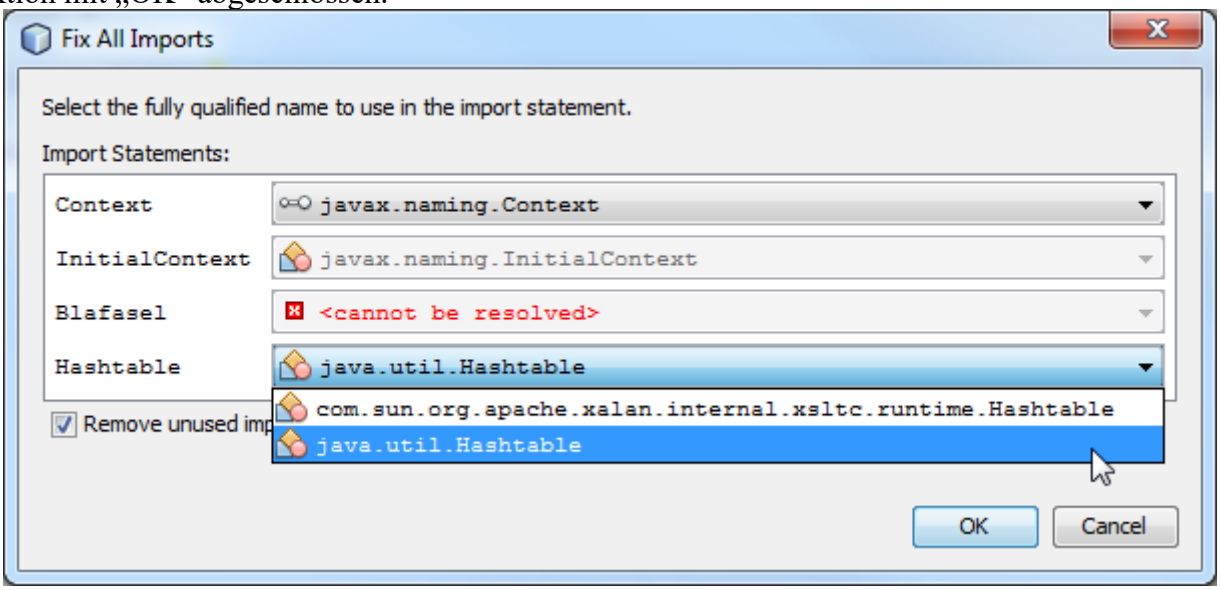
```
14      public static void main(String[] args) {
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY
17                  , "com.sun.jndi.fscontext.RefFSContext");
            Context context = new InitialContext(env);
            Blafasel b;
20      }
```

Mit einem Rechtsklick im Editor kann man „Fix Import“ auswählen.



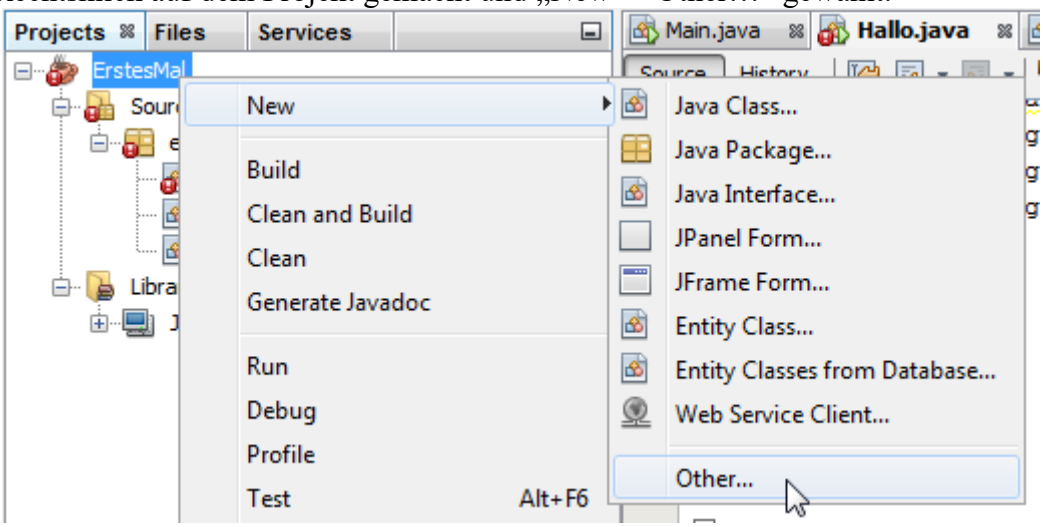
Dabei. muss man drei Situationen unterscheiden. Es gibt keine passende Klasse in den Java-Bibliotheken und dem bisherigen Programm (Blafasel), es gibt genau eine passende Klasse (InitialContext) oder es gibt mehrere passende Klassen (HashTable). Gerade die letzte Situation ist gefährlich für unerfahrene Nutzer, da es typischerweise nur eine richtige Klasse gibt. Das

Auswahlmenü wird durch einen Klick auf den Pfeil nach unten am rechten Rand aufgeklappt und die Aktion mit „OK“ abgeschlossen.

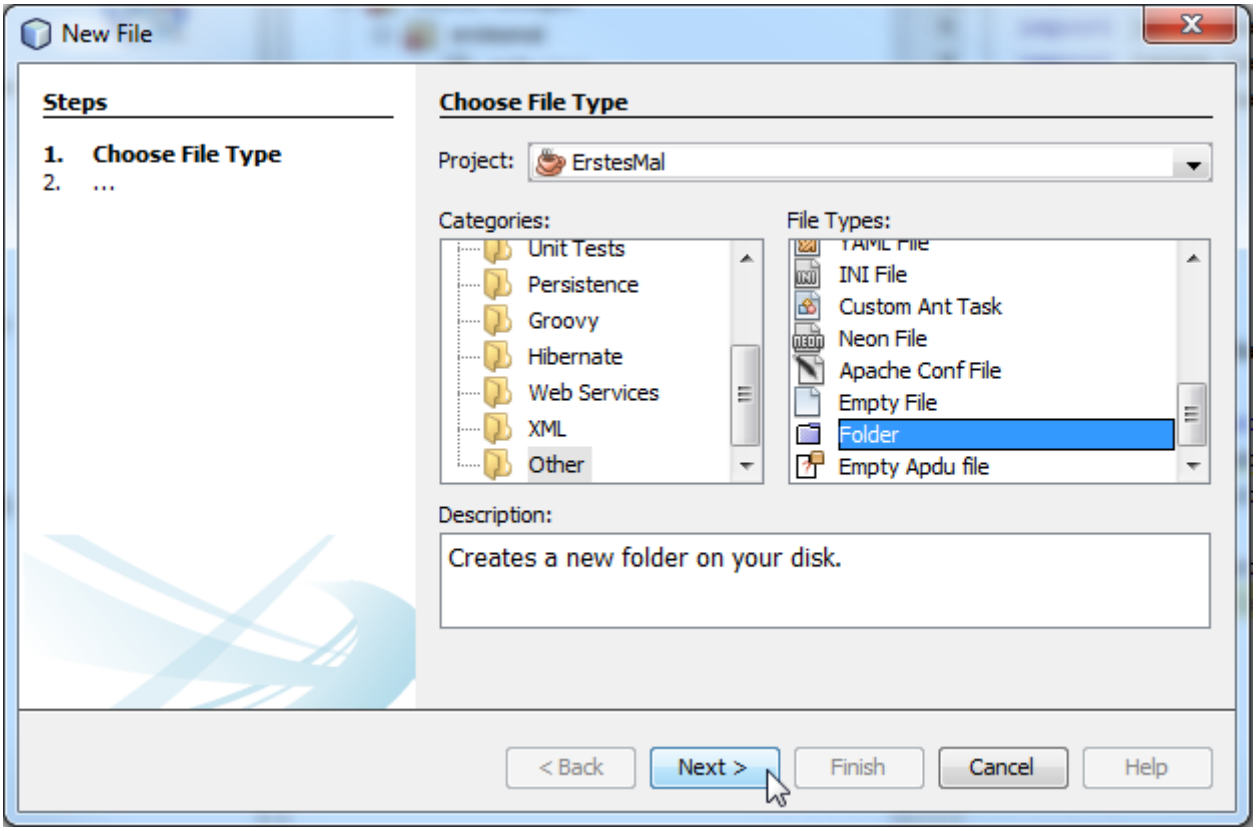


4.6 Nutzung von Bibliotheken/Jar-Files (NetBeans 8)

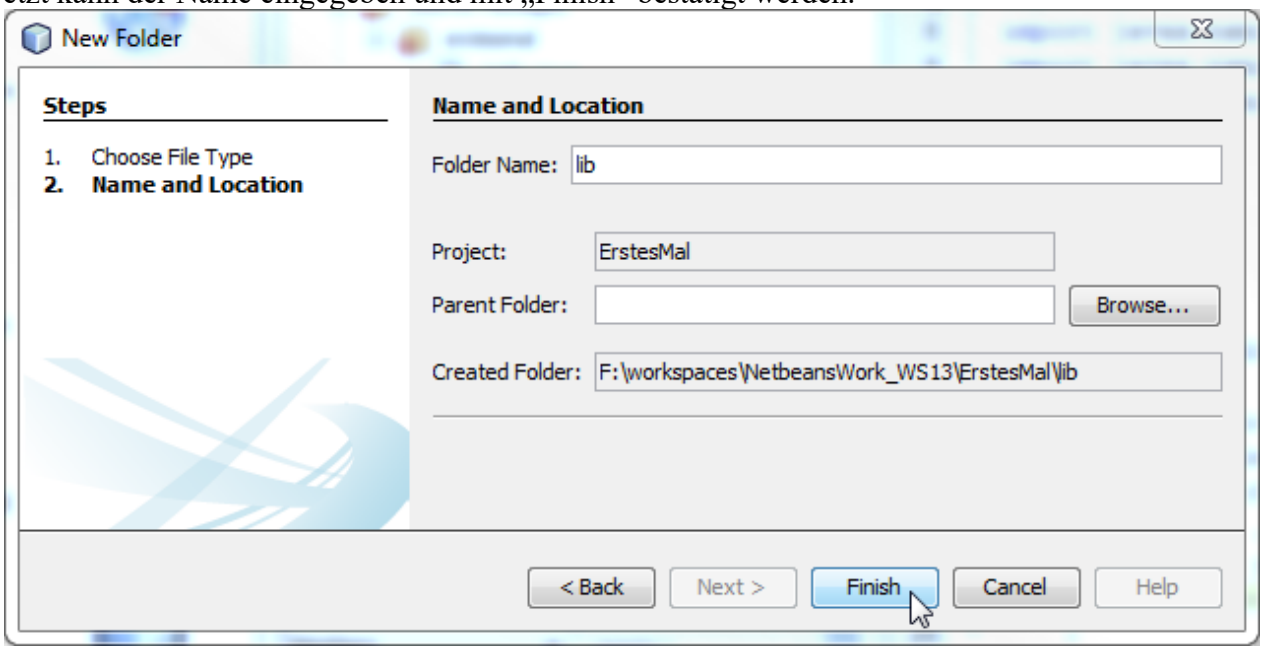
Möchte man Bibliotheken (Jar-Dateien) nutzen, müssen diese über den CLASSPATH zur Verfügung stehen. Zur Nutzung von Jar-Dateien, sollten diese sich zentral in einem Ordner auf dem Rechner oder besser noch in jedem Projekt individuell befinden, wodurch es keine Konflikte geben kann, wenn unterschiedliche Versionen der gleichen Bibliothek benötigt werden. Dazu wird zunächst ein Ordner lib angelegt. Dies könnte natürlich auch direkt im Projektordner erledigen, da man generell solche Ordner möglichst wenig ohne NetBeans bearbeiten soll, wird hier ein etwas längerer Weg genutzt. Es wird ein Rechtsklick auf dem Projekt gemacht und „New -> Other…“ gewählt.



Es wird unten auf der linken Seiter wieder „Other…“ und rechts „Folder“ ausgewählt und „Next>“ geklickt.

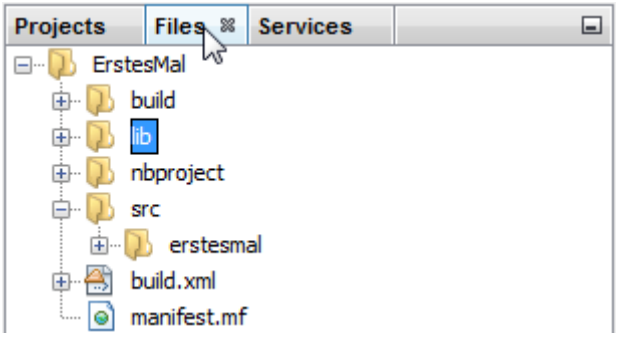


Jetzt kann der Name eingegeben und mit „Finish“ bestätigt werden.

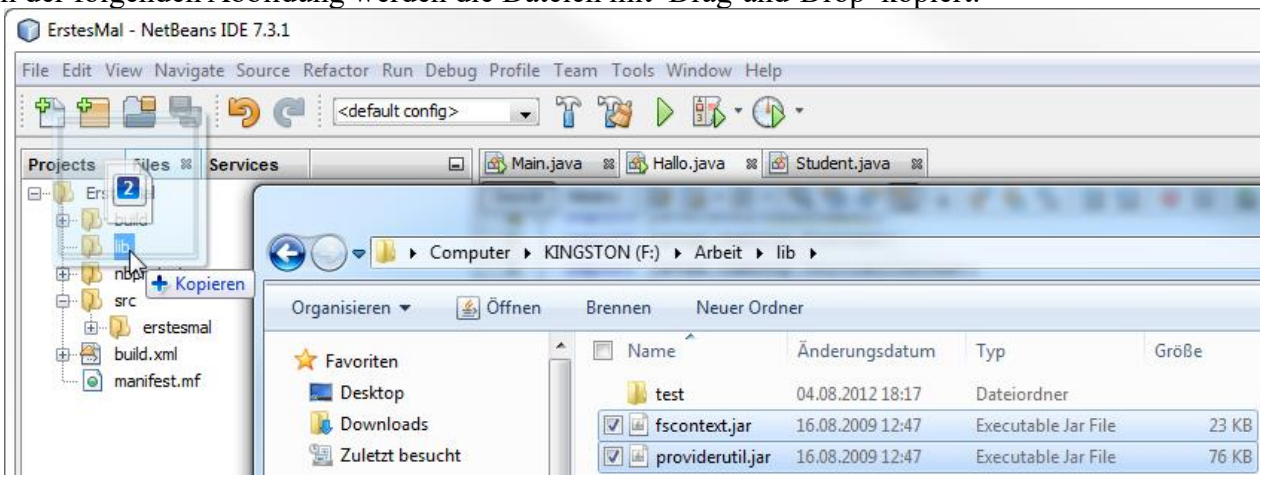


Der lib-Ordner ist wieder auf sehr unterschiedliche Arten befüllbar. Eine Variante ist es die jar-Dateien direkt per Drag-and-Drop auf das lib-Verzeichnis zu ziehen. Dazu muss oben links im Projekt-Browser zunächst „Files“ angewählt werden. Man sieht dort den entstandenen lib-Ordner,

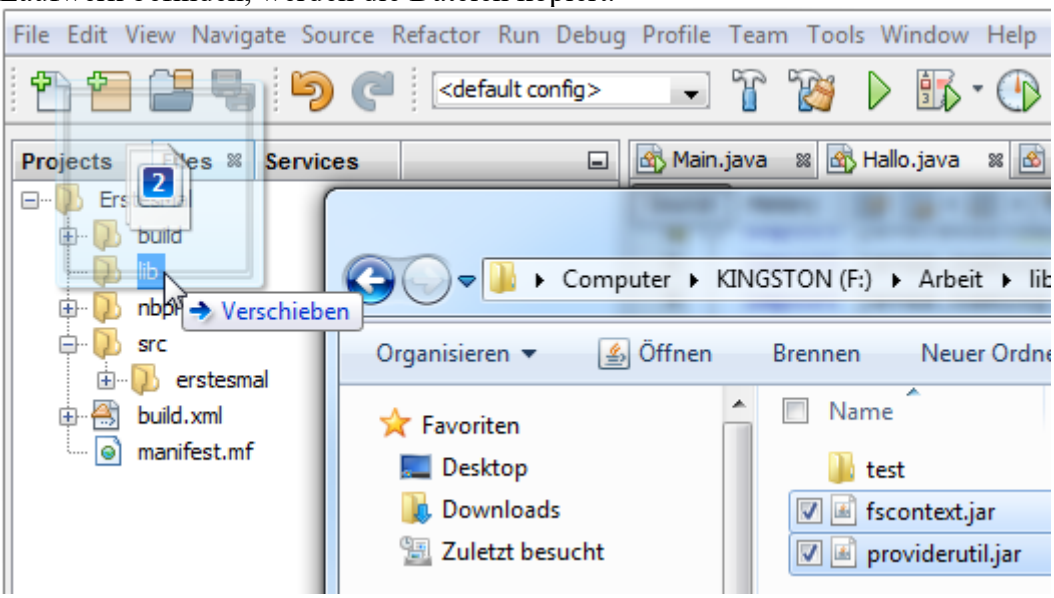
der auch hier hätte erzeugt werden können.



In der folgenden Abbildung werden die Dateien mit Drag-and-Drop kopiert.

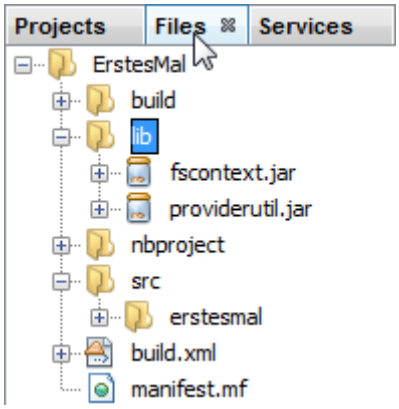


Auch wenn die Meldung „Verschieben“ angezeigt wird, da sich Quelldateien und Zielordner auf dem gleichen Laufwerk befinden, werden die Dateien kopiert.

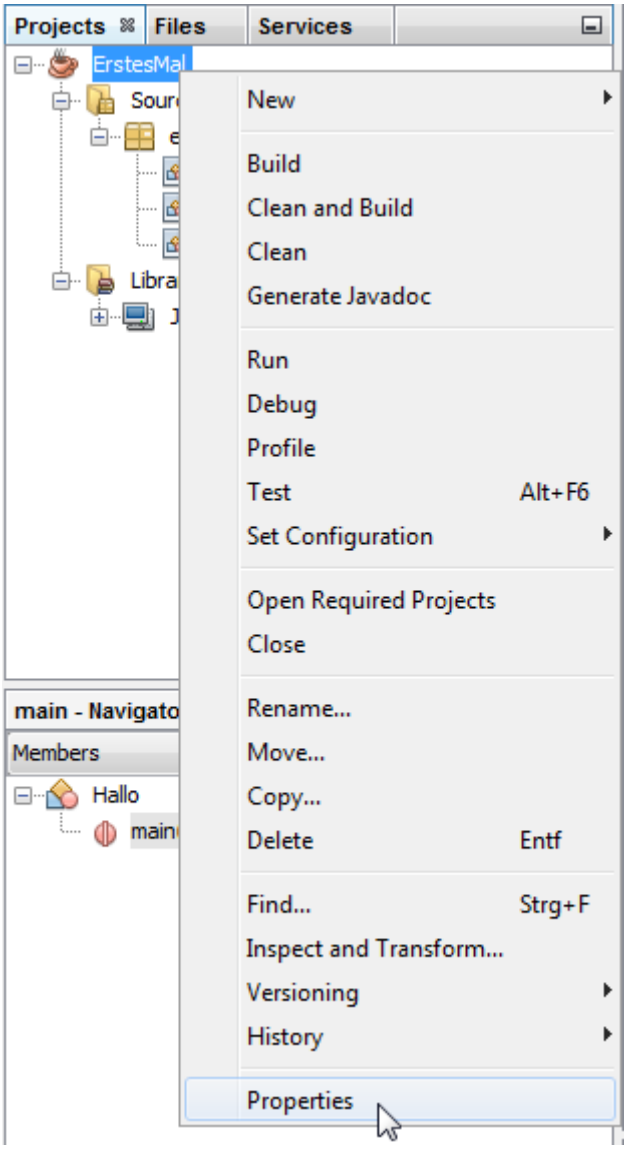


Nun befinden sich die Bibliotheken im Unterordner lib, sind allerdings noch nicht in das Java-Projekt

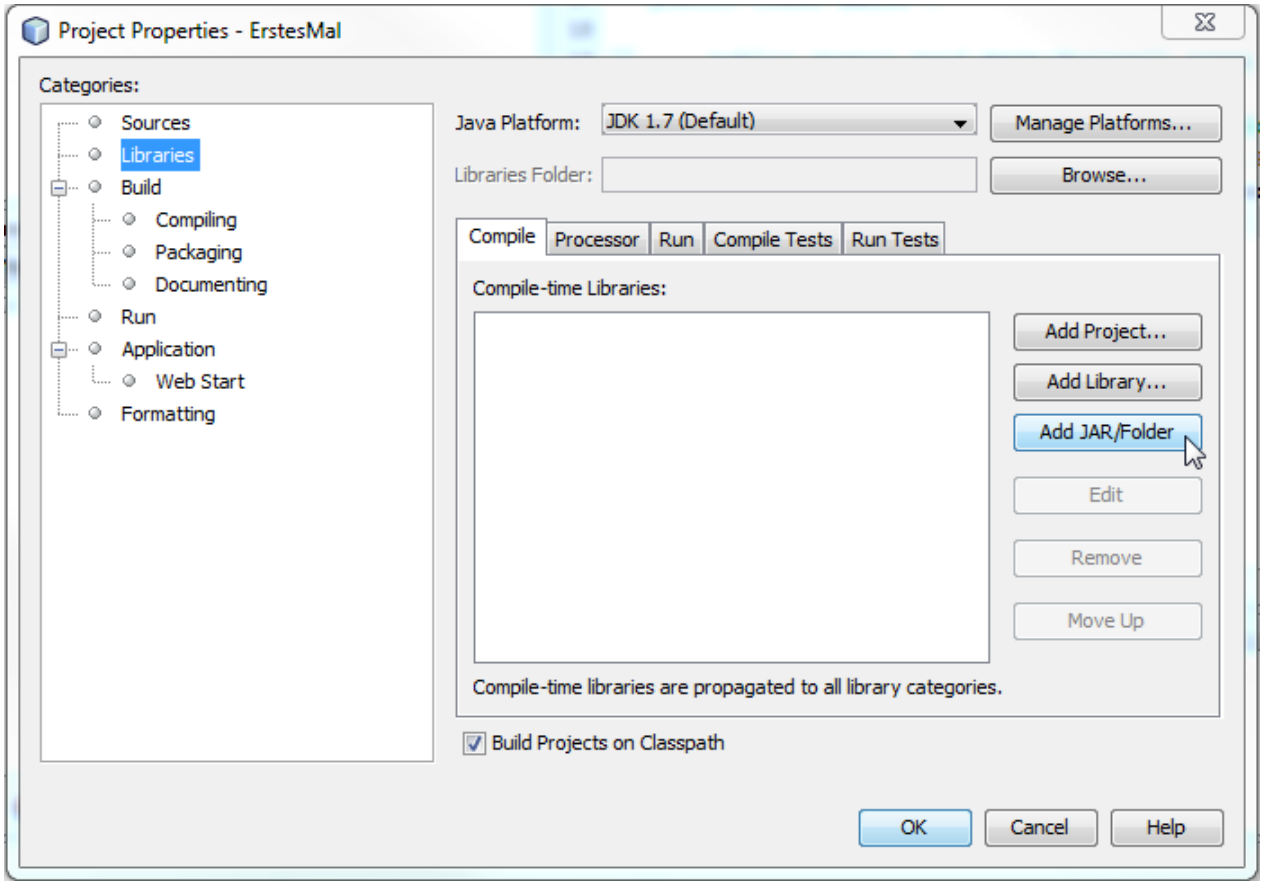
eingebunden.



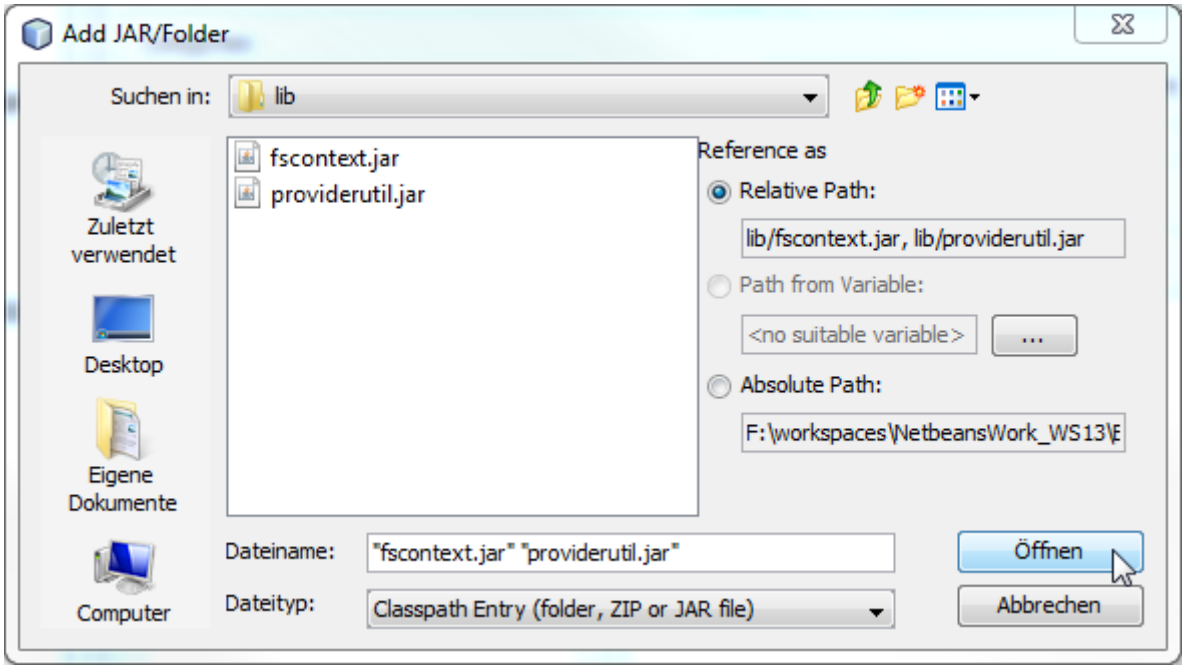
In NetBeans unter der Auswahl „Projects“ kann man dann projektindividuell Bibliotheken ergänzen. Dazu wird ein Rechtsklick auf das Projekt gemacht und unten „Properties“ ausgewählt.



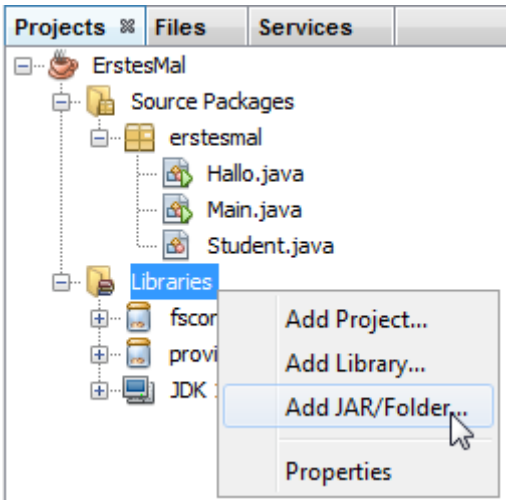
Danach wird links „Libraries“ gewählt und es stehen rechts einige interessante Alternativen zur Verfügung. Mit „Add Project...“ kann man Beziehungen zu anderen Projekten aufbauen, deren Ergebnisse möglichst als Jar-Dateien vorliegen sollten. Mit „Add Library...“ kann man Bibliotheken auswählen, die bereits einfach in NetBeans vorhanden sind, der Punkt „Create...“ führt dabei zu einem Zusammenstellungswerkzeug, mit dem man solch eine Zusammenfassung von Jar-Dateien selbst erstellen kann (hier nicht weiter betrachtet).
Möchte man Jar-Dateien nutzen, sollten sie sich in einem Ordner auf dem Rechner befinden. Es sollen hier jar-Dateien aus dem vorher angelegten Ordner eingebunden werden. Zum Einbinden klickt man auf „Add JAR/Folder“.



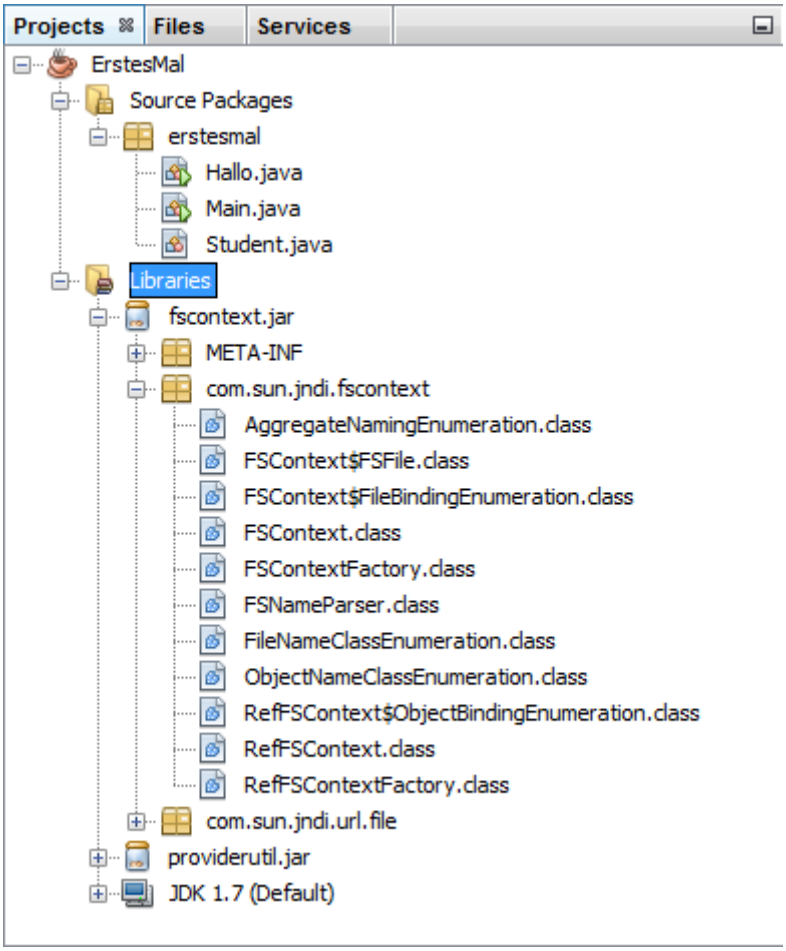
Danach wird in das Verzeichnis mit den Jar-Dateien gesteuert. Mehrere Jar-Dateien können durch das Festhalten der <Strg>-Taste ausgewählt werden. Am rechten Rand kann bestimmt werden, ob der Zugriff einen relativen oder einen absoluten Pfad nutzt. Absolute Pfade haben den wesentlichen Nachteil, dass auf einem anderen Rechner, der die entstehende Software nutzen will, exakt die gleichen Pfade existieren müssen. Deshalb empfiehlt es sich in den meisten Fällen, relative Pfade zu nutzen, weshalb es auch sinnvoll ist, die genutzten Jar-Dateien in ein lib-Verzeichnis des aktuellen Projekts zu legen, was so verschiebbar wird. Mit „Öffnen“ werden die Bibliotheken hinzugefügt.



Ein alternativer Weg zur Einbindung von Bibliotheken entsteht, wenn man einen Rechtsklick auf den Unterordner Libraries im Projektordner macht. Man kommt über „Add JAR/Folder...“ auch zum vorherigen Auswahlfenster.

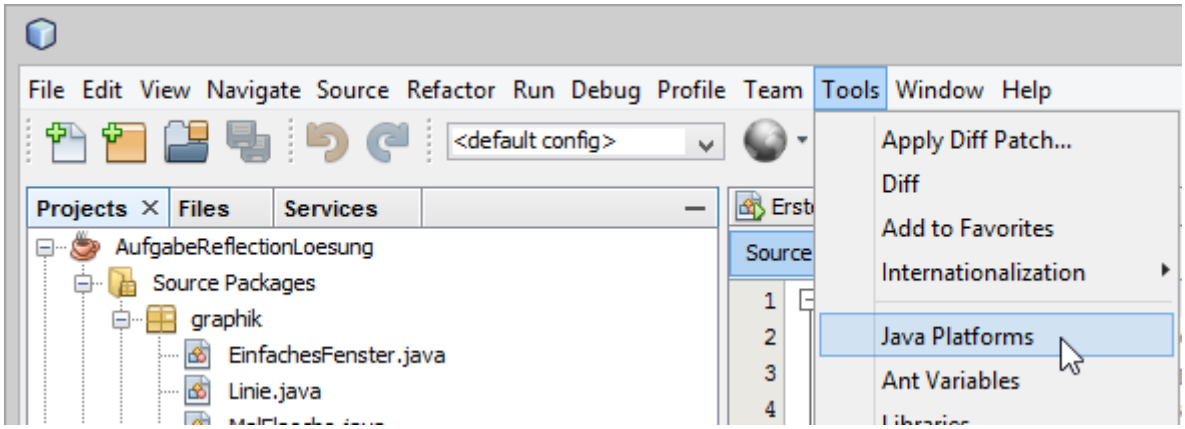


Das Ergebnis sieht wie folgt aus, wobei man durch Aufklappen der Jar-Dateien Informationen über ihre Inhalte erhält.

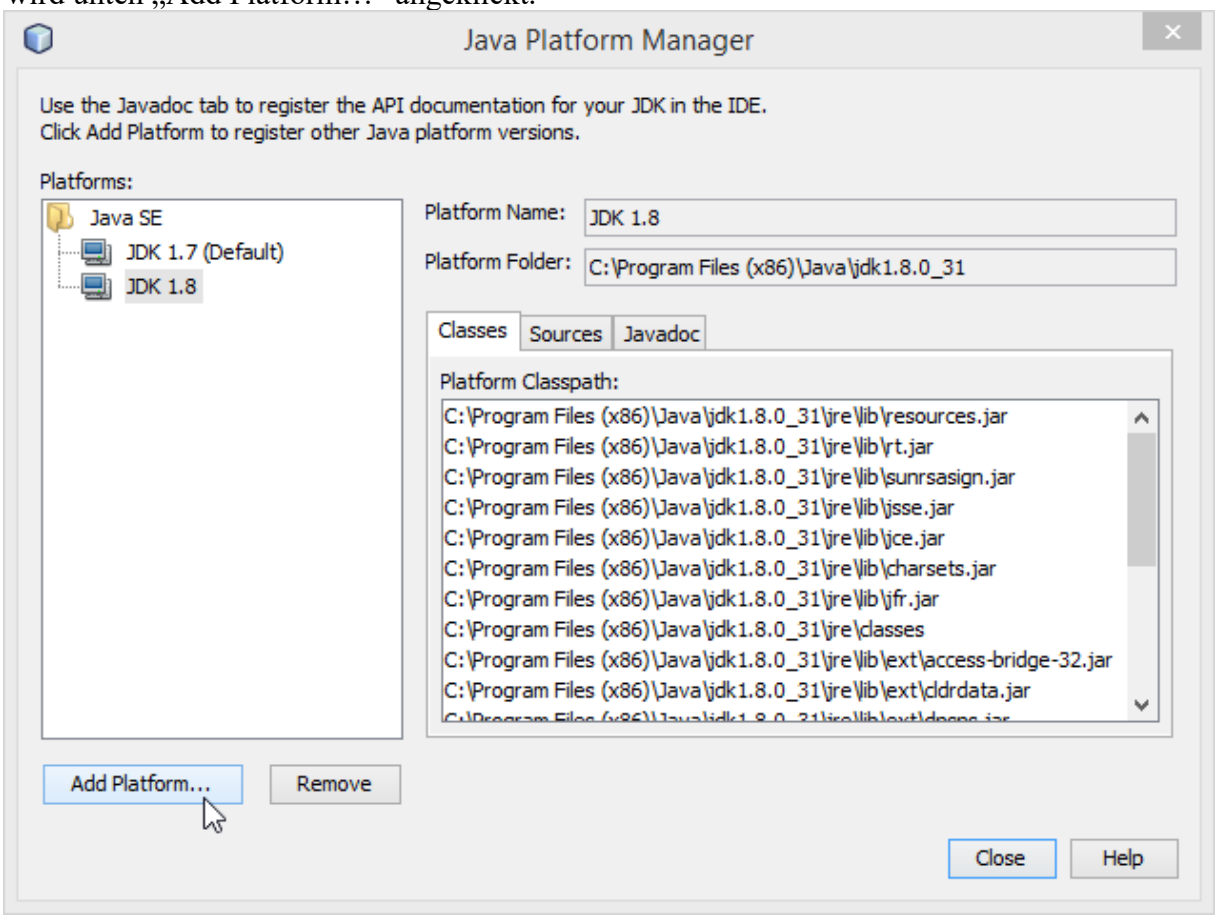


4.7 Installation einer neueren/weiteren Java-Version

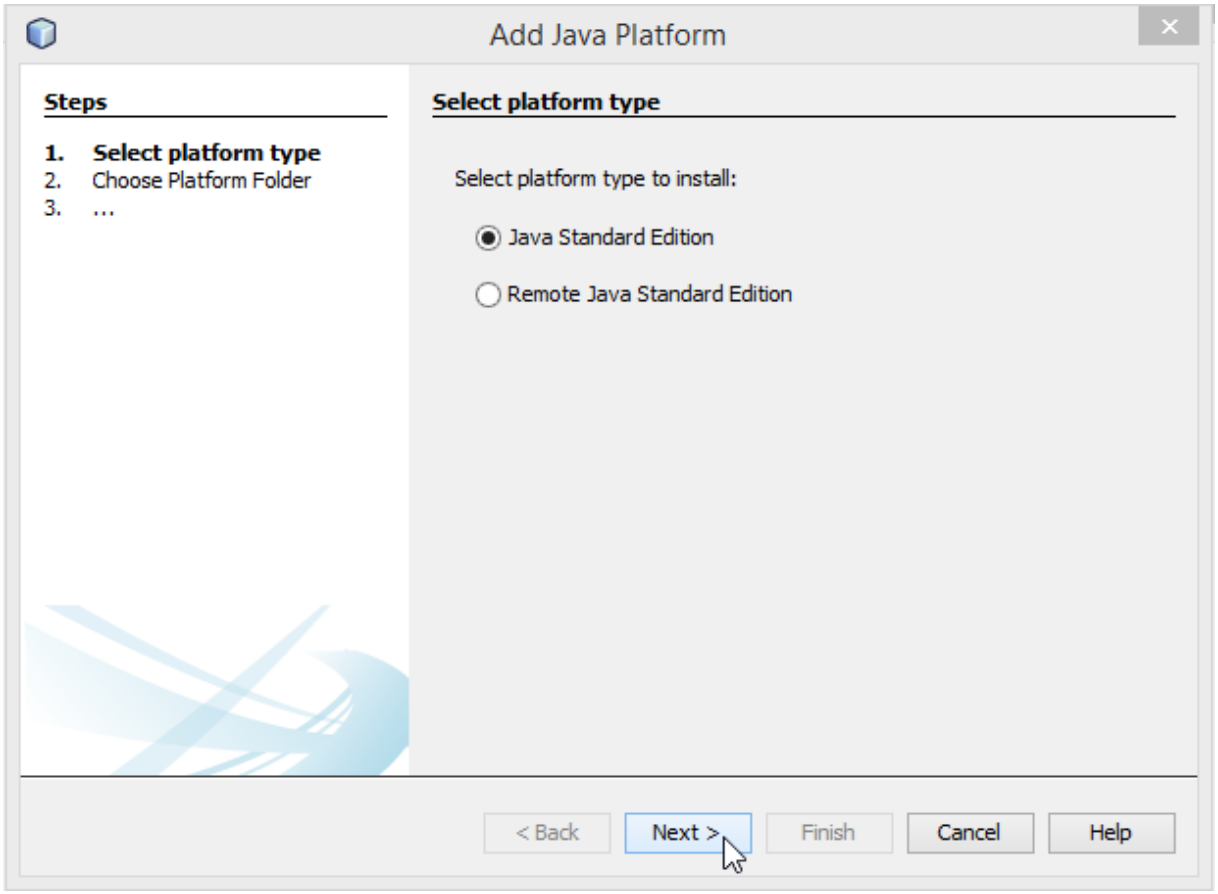
Generell besteht keine echte Notwendigkeit die Java-Version kontinuierlich zu aktualisieren, da sich im Internet genannte Gefahren (fast) ausschließlich auf die Nutzung von Java im Browser beziehen. Trotzdem kann es sinnvoll sein, eine neue Java-Version zu ergänzen, u. z. B. neue Bibliotheken nutzbar zu machen. In diesem Unterkapitel wird davon ausgegangen, dass die neue Java-Version bereits auf dem Rechner installiert ist und einem bereits vorher genutzten NetBeans zugänglich gemacht werden soll.
Die Aktualisierung oder genauer Ergänzung einer JDK-Version zur Nutzung in NetBeans startet mit Klicks auf „Tools -> Java Platforms“.



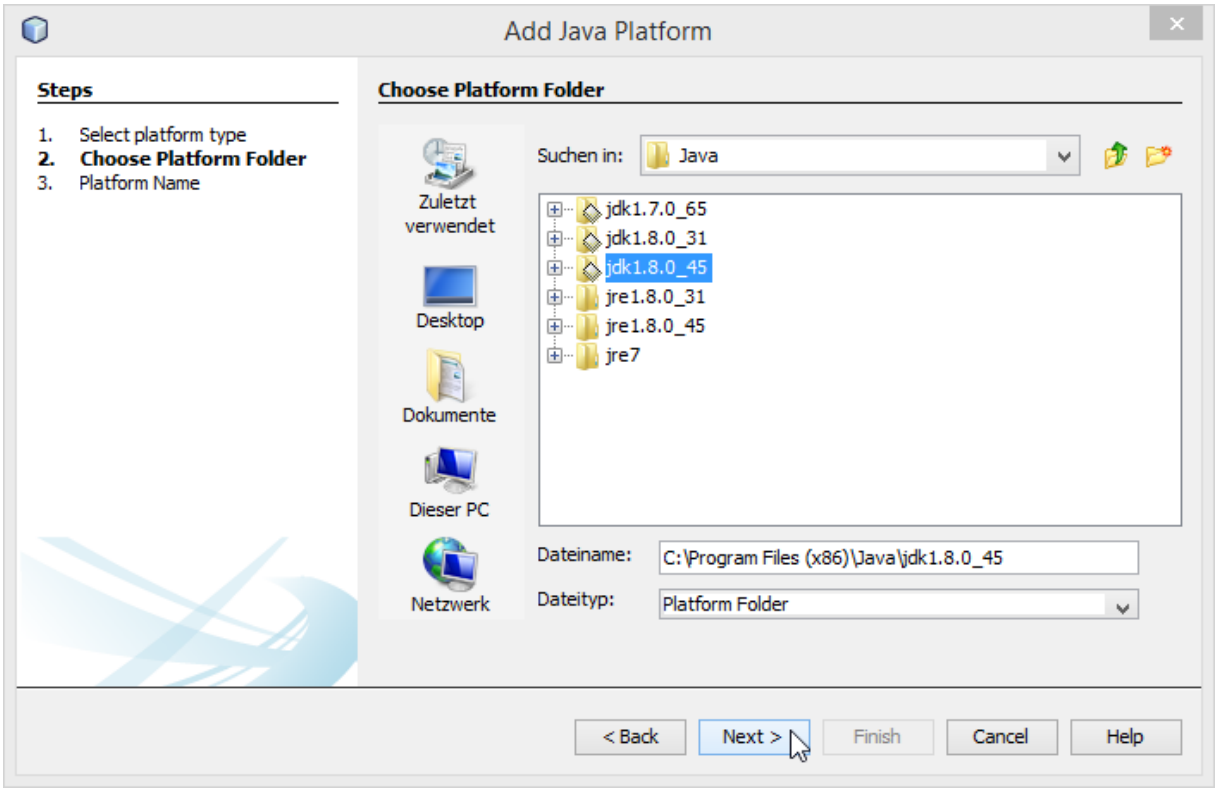
Es wird unten „Add Platform…“ angeklickt.



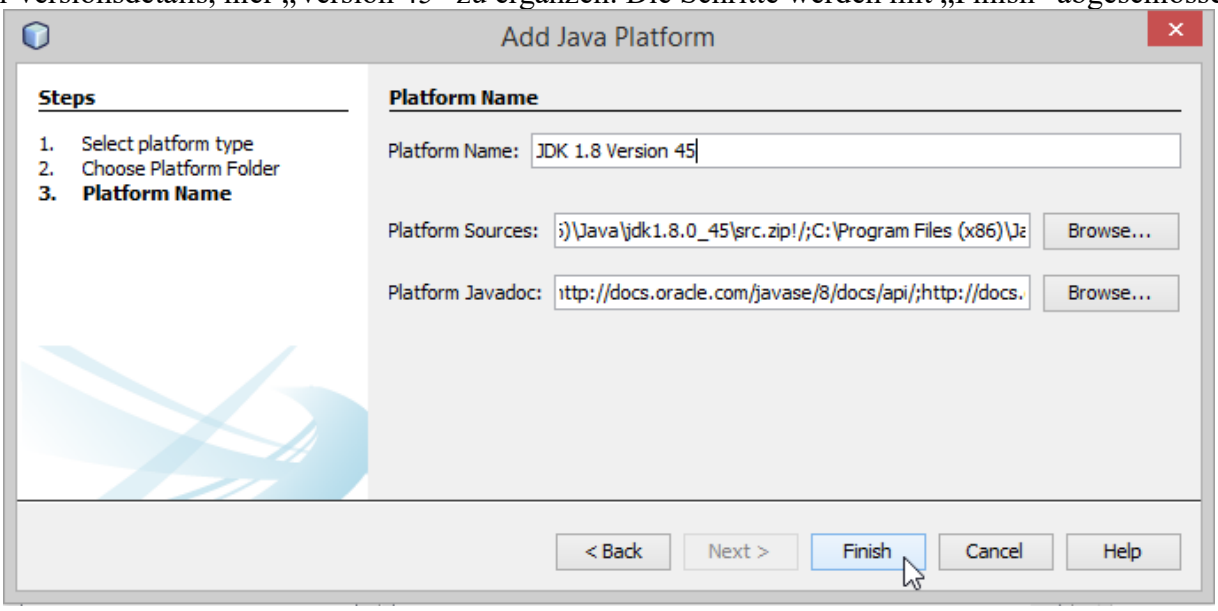
Es wird „Next >“ geklickt.



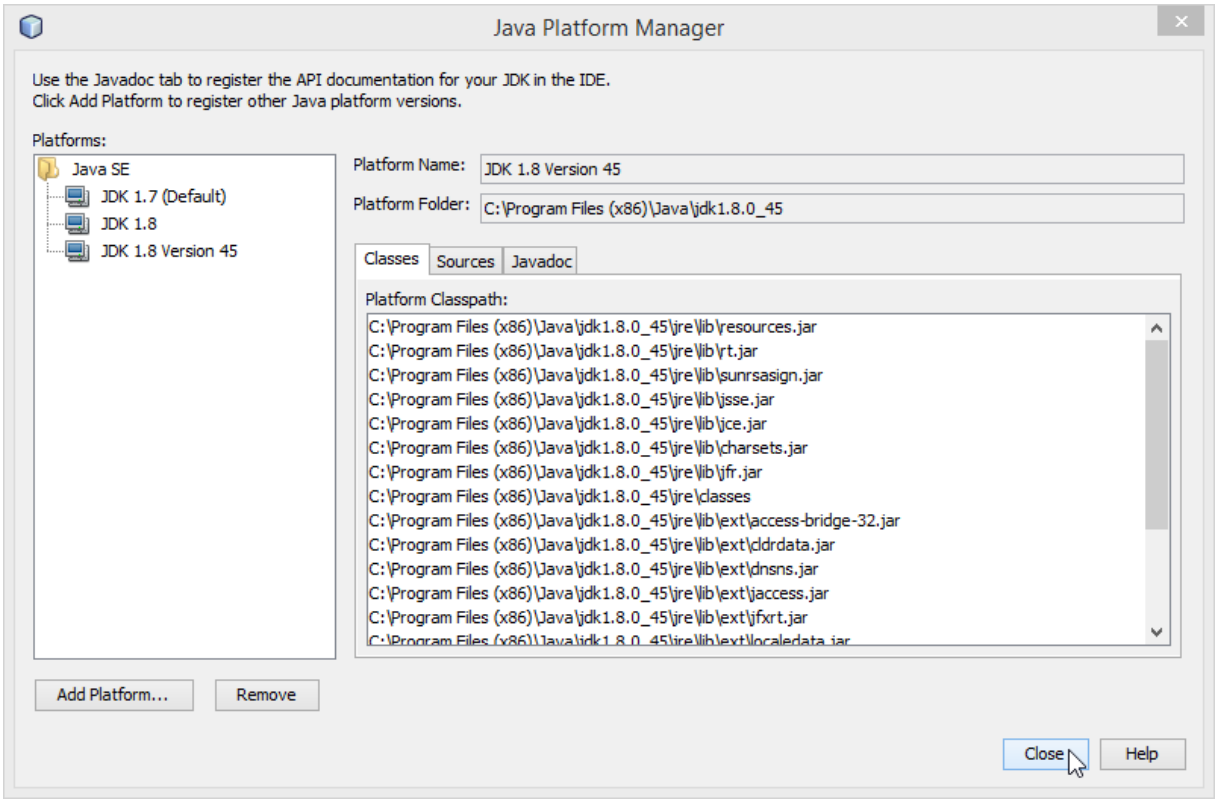
Es wird zum Verzeichnis mit der neuen Installation gesteuert. Hier wird das mit „jdk“ beginnende Verzeichnis ausgewählt und „Next >“ geklickt.



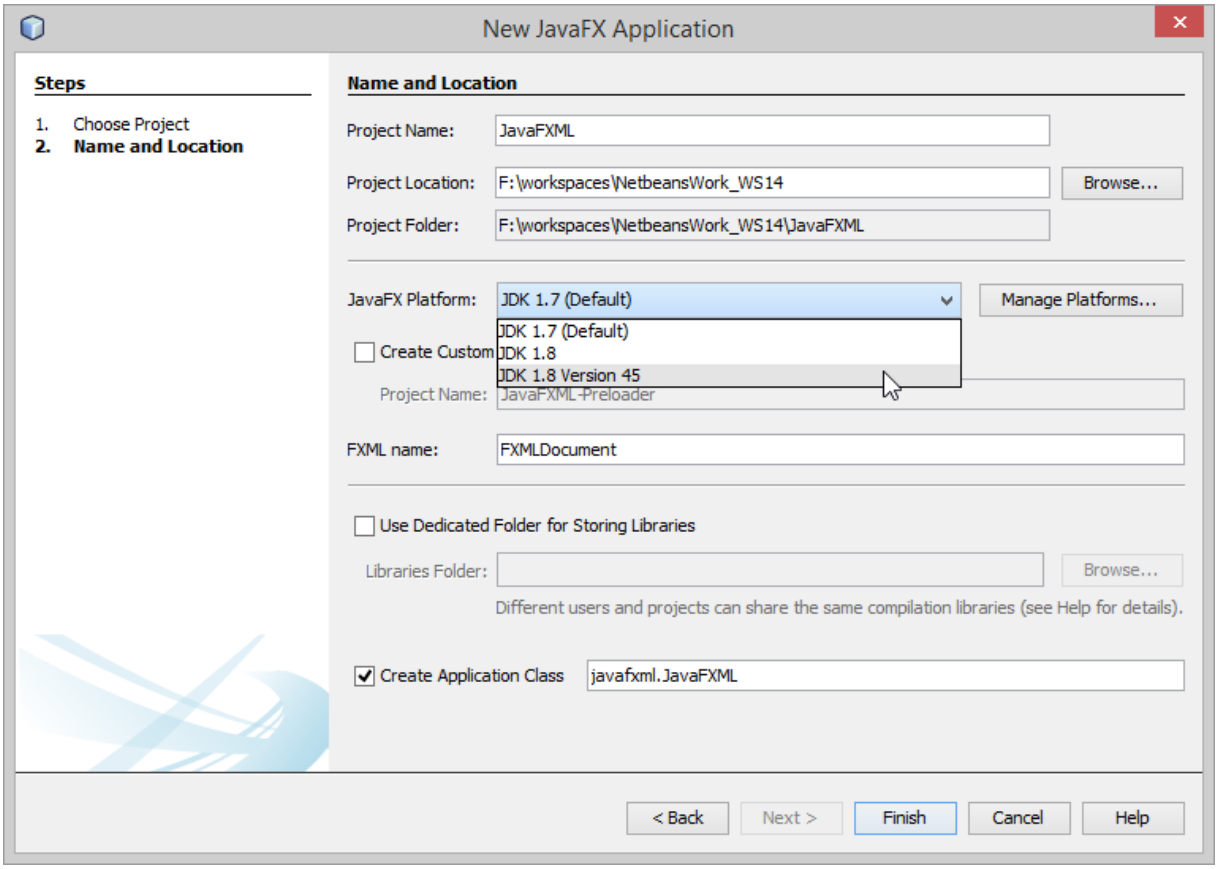
Die Datenfelder werden automatisch gefüllt. Generell ist es sinnvoll, den oberen „Platform Name“ um Versionsdetails, hier „Version 45“ zu ergänzen. Die Schritte werden mit „Finish“ abgeschlossen.



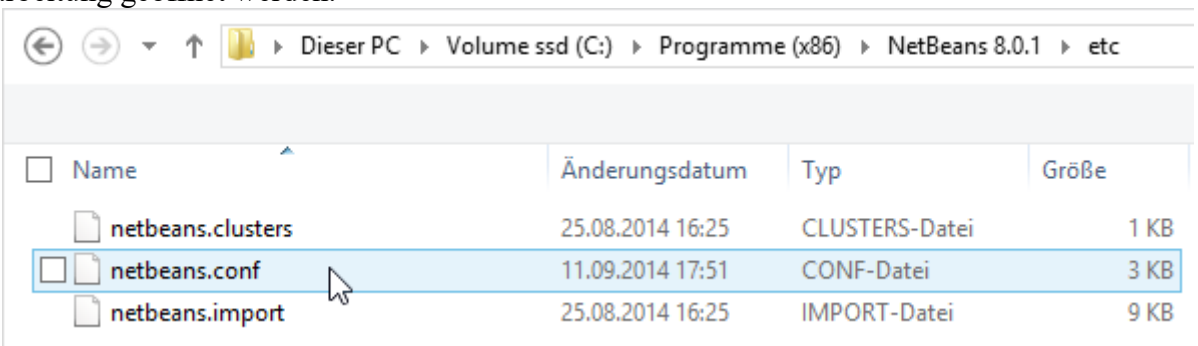
Der Abschluss der Ergänzung erfolgt mit „Close“.



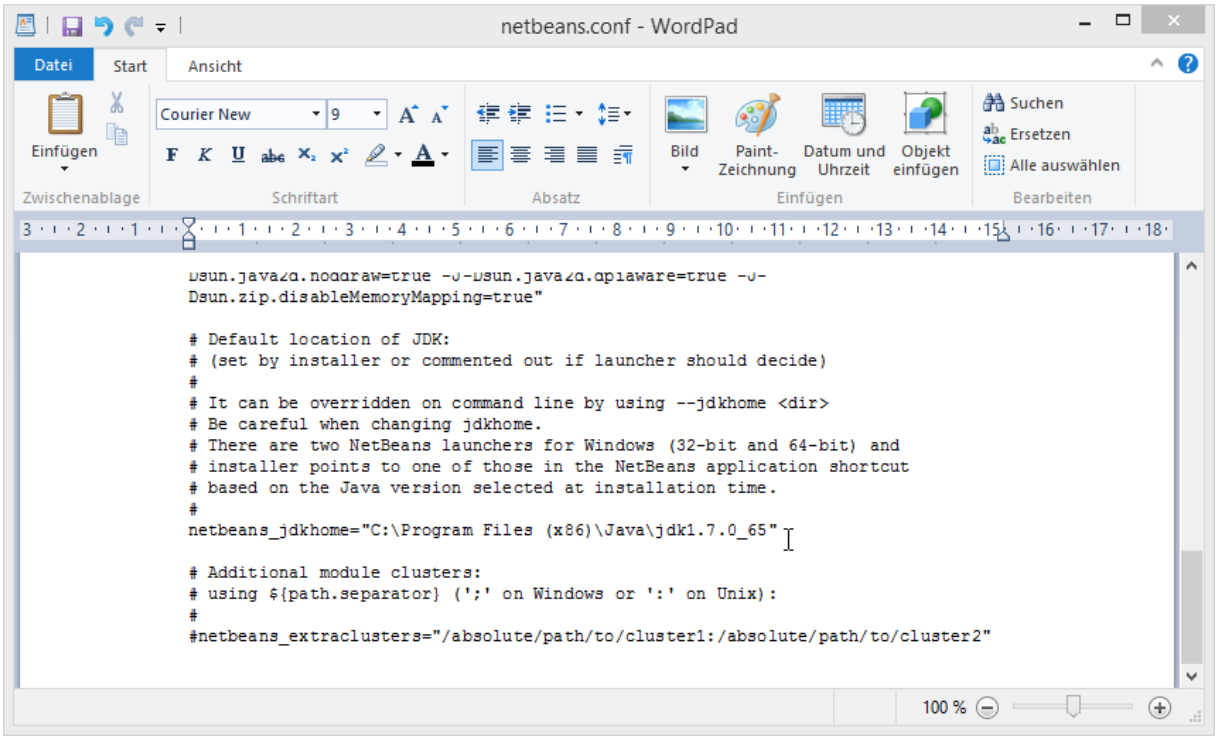
Nun kann bei der Erstellung eines neuen Projekts das gewünschte JDK ausgewählt werden.



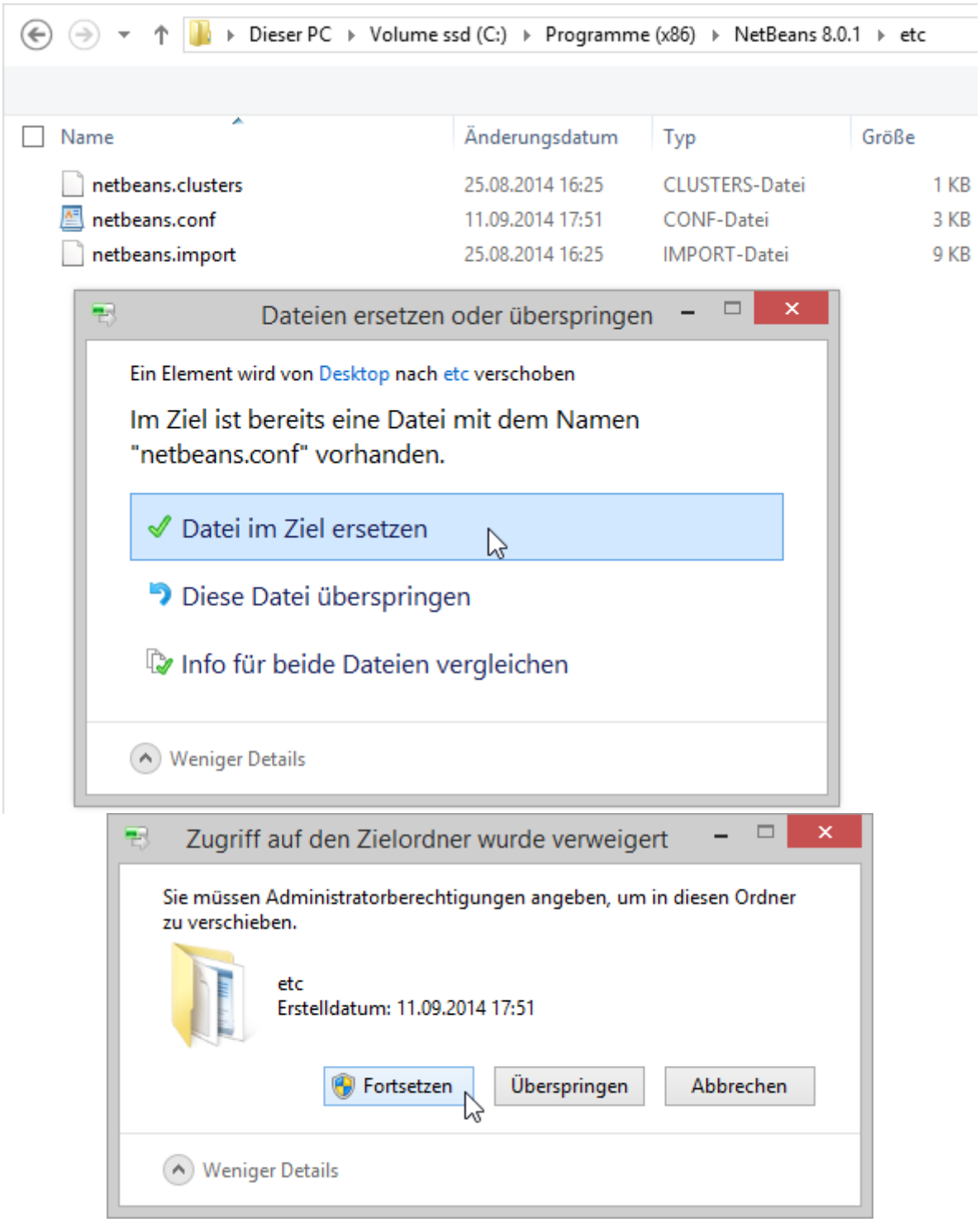
Möchte man die Default-Version insgesamt ändern, was sich dann auf alle genutzten NetBeans-Projekte auswirkt, muss in der NetBeans-Installation die Datei etc/netbeans.conf gesucht und zur Bearbeitung geöffnet werden.



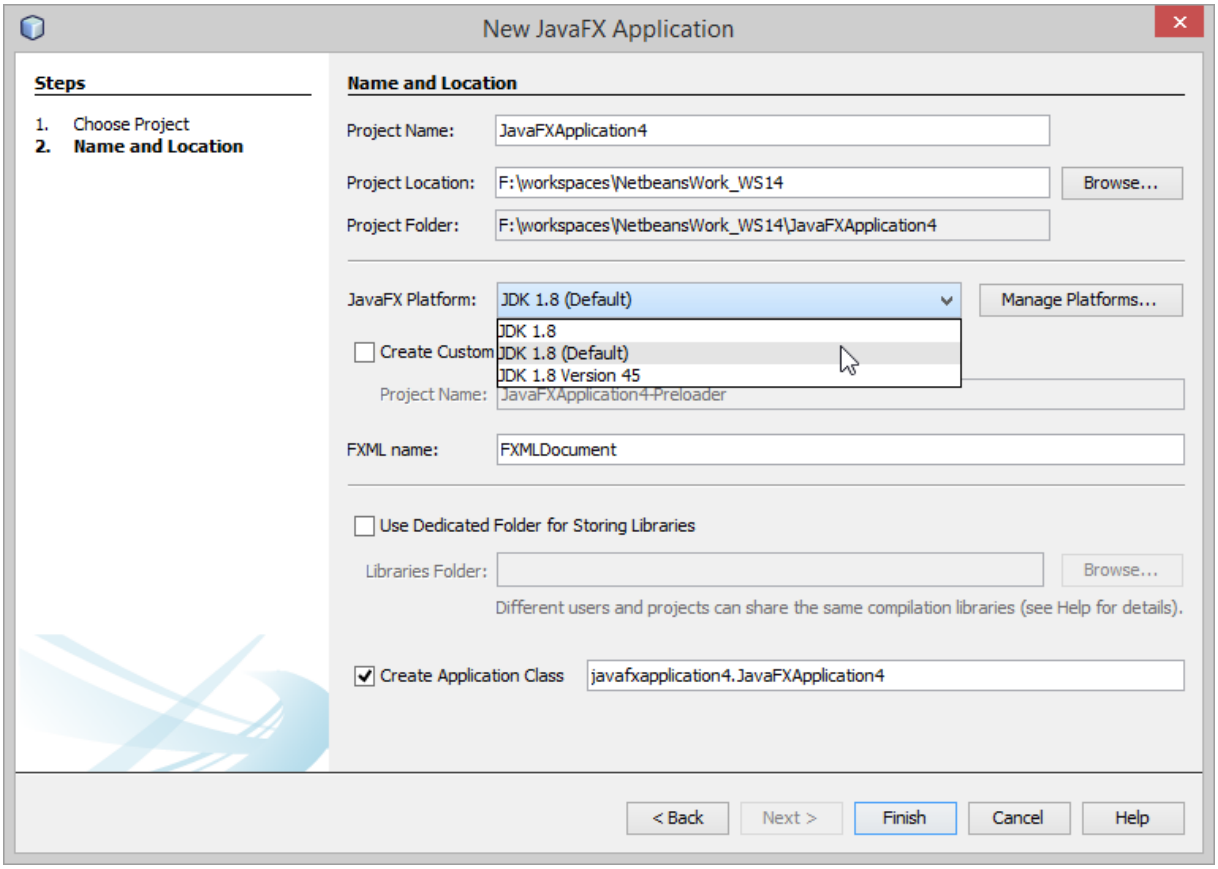
Unten in der Datei befinden sich Informationen zur „Default location of JDK“. Hier muss dann die Variable netbeans_jdkhome passend geändert werden. Falls man direkt keine Schreibrechte hat, wird die Datei in einem anderen Ordner abgespeichert und dann in den Zielordner kopiert.



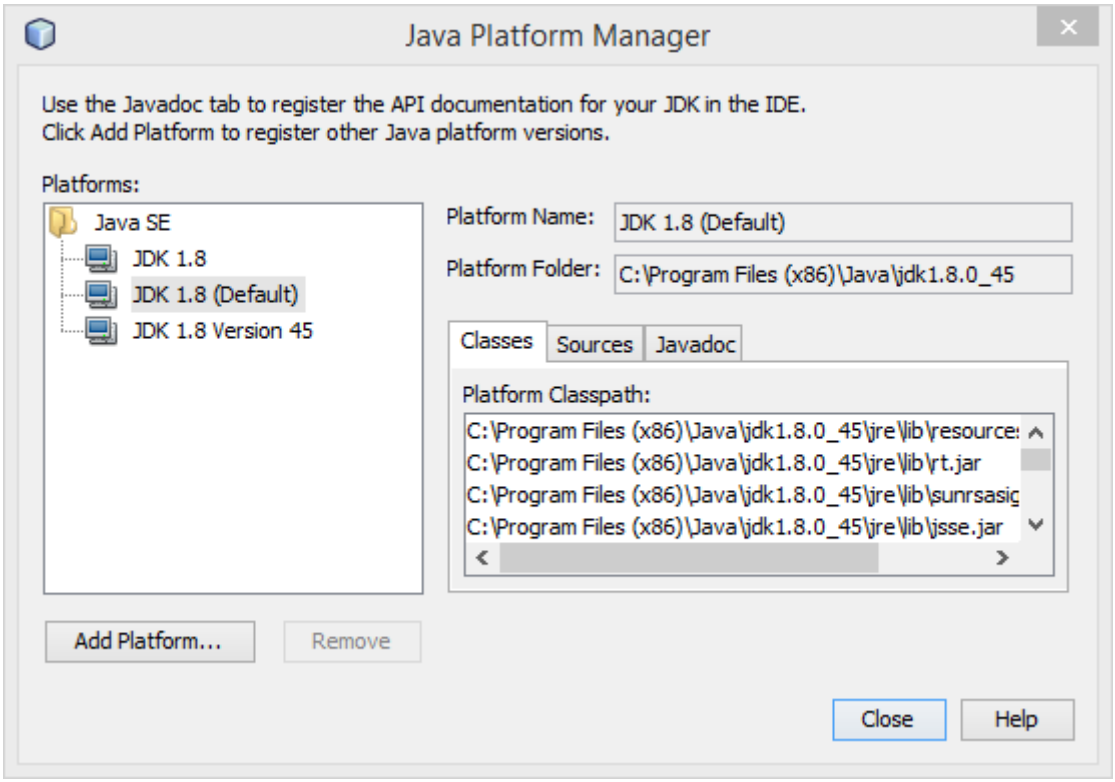
Beim Verschieben der veränderten Datei in den etc-Ordner muss dem Überschreiben zugestimmt werden.



Die Änderung wird nach einem Neustart von NetBeans aktiv. Die Platform-Auswahl kann dann durch den neuen Default-Wert etwas unübersichtlich werden.



Im „Platform Manager“ wird deutlich, dass die Umstellung erfolgreich war.


Java Platform Manager
Use the Javadoc tab to register the API documentation for your JDK in the IDE.
Click Add Platform to register other Java platform versions.
Platforms:
Java SE
JDK 1.8
JDK 1.8 (Default)
JDK 1.8 Version 45
Platform Name:
JDK 1.8 (Default)
Platform Folder:
C:\Program Files (x86)\Java\jdk1.8.0_45
Classes
Sources
Javadoc
Platform Classpath:
C:\Program Files (x86)\Java\jdk1.8.0_45\jre\lib\resource
C:\Program Files (x86)\Java\jdk1.8.0_45\jre\lib\rt.jar
C:\Program Files (x86)\Java\jdk1.8.0_45\jre\lib\sunrsasig
C:\Program Files (x86)\Java\jdk1.8.0_45\jre\lib\jsse.jar
Add Platform...
Remove
Close
Help

5 Nutzung von JUnit (NetBeans 8)

In einer systematischen Software-Entwicklung müssen Programme systematisch getestet werden. Auf Entwicklungsebene werden dazu sogenannte Unit-Testwerkzeuge genutzt. JUnit ist dabei ein Beispiel, wobei JUnit auch für viele weitere Testmöglichkeiten nutzbar ist. JUnit ist direkt in NetBeans integriert, ist man den Installationsanleitungen gefolgt, wird es genauer mit dem ersten Start von NetBeans nachinstalliert. Es gibt verschiedene Möglichkeiten JUnit einzubinden, hier werden einzelne davon gezeigt.

5.1 Vollständige Nutzung der JUnit-NetBeans-Unterstützung

Der erste Ansatz zeigt, wie JUnit direkt von NetBeans unterstützt wird. Mit etwas Erfahrungen können auch andere Wege gewählt werden, die Inhalt der nachfolgenden Unterkapitel sind.
Es wird davon ausgegangen, dass zunächst ein Programm entstanden ist, das getestet werden soll, wobei auch der Weg erst Tests zu schreiben und dann das Programm zu realisieren sinnvoll ist. Im Ausgangsprojekt sind noch keine Tests vorhanden.

```
package entity;

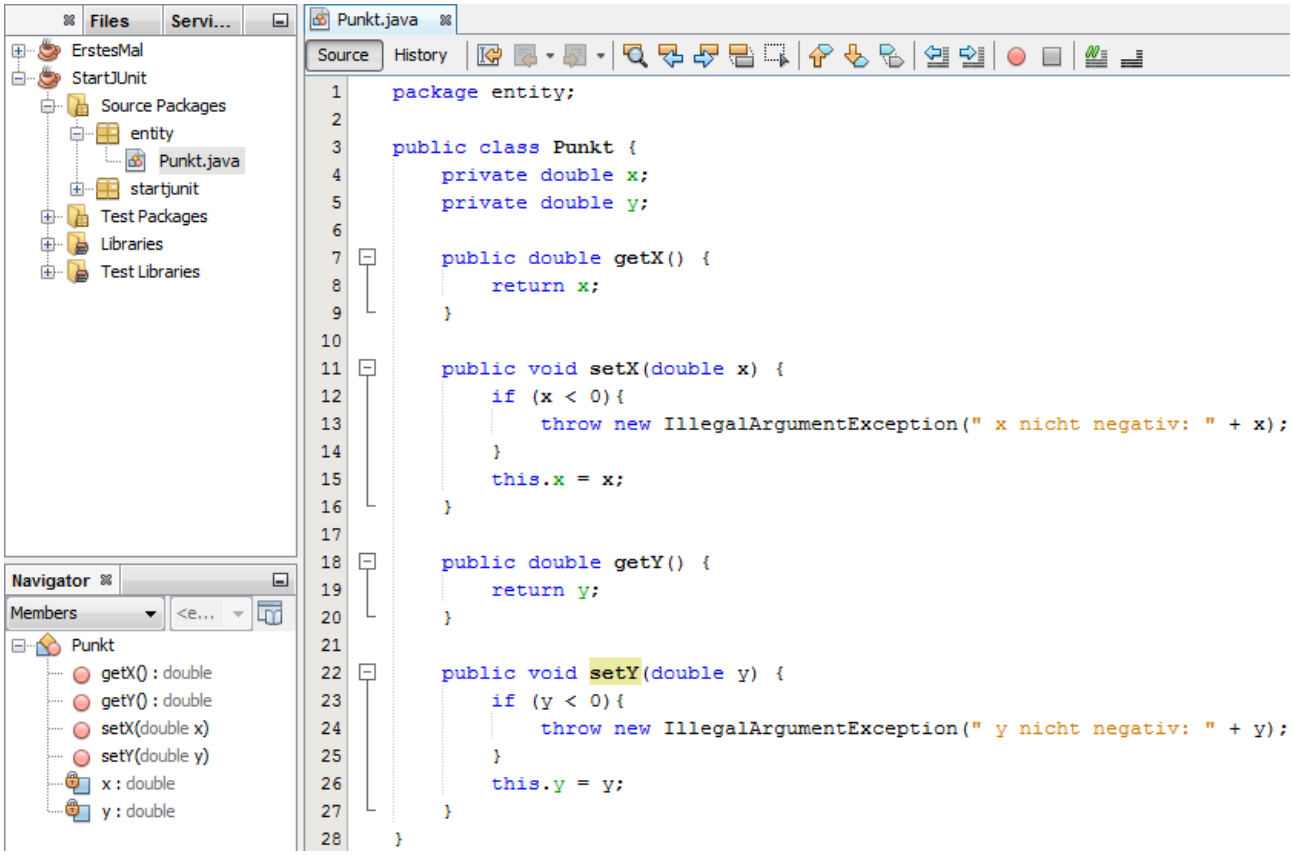
public class Punkt {
    private double x;
    private double y;

    public double getX() {
        return x;
    }

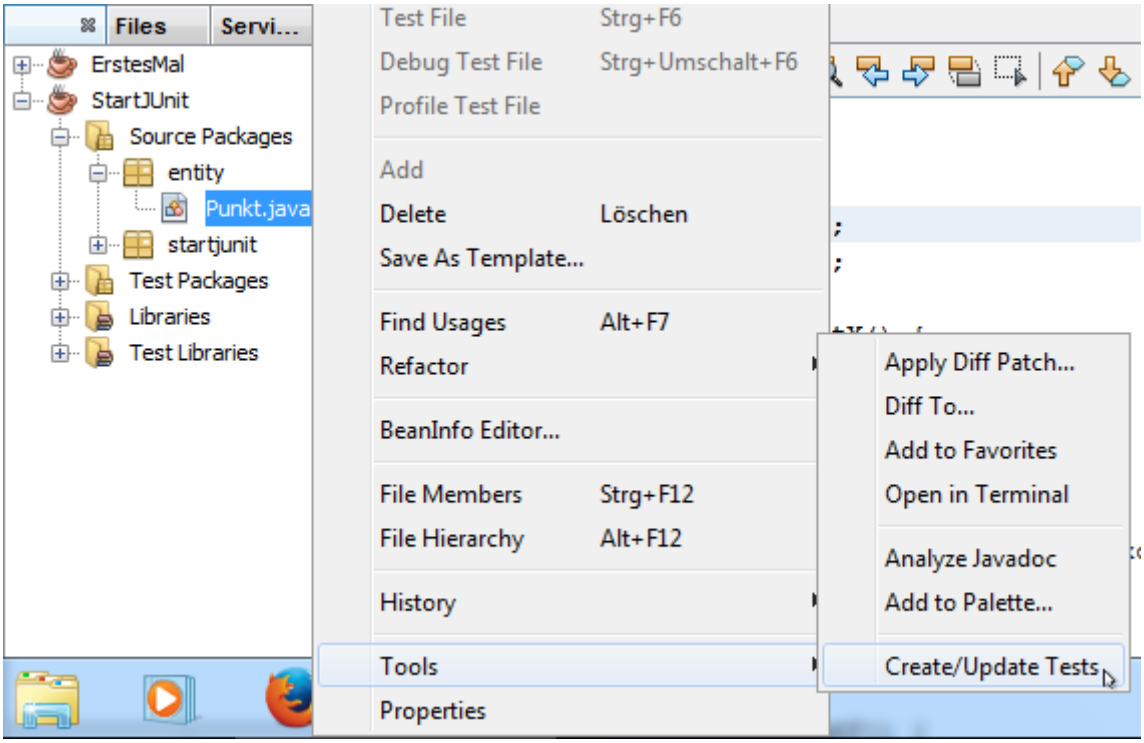
    public void setX(double x) {
        if (x < 0){
            throw new IllegalArgumentException(" x nicht negativ: " + x);
        }
        this.x = x;
    }

    public double getY() {
        return y;
    }

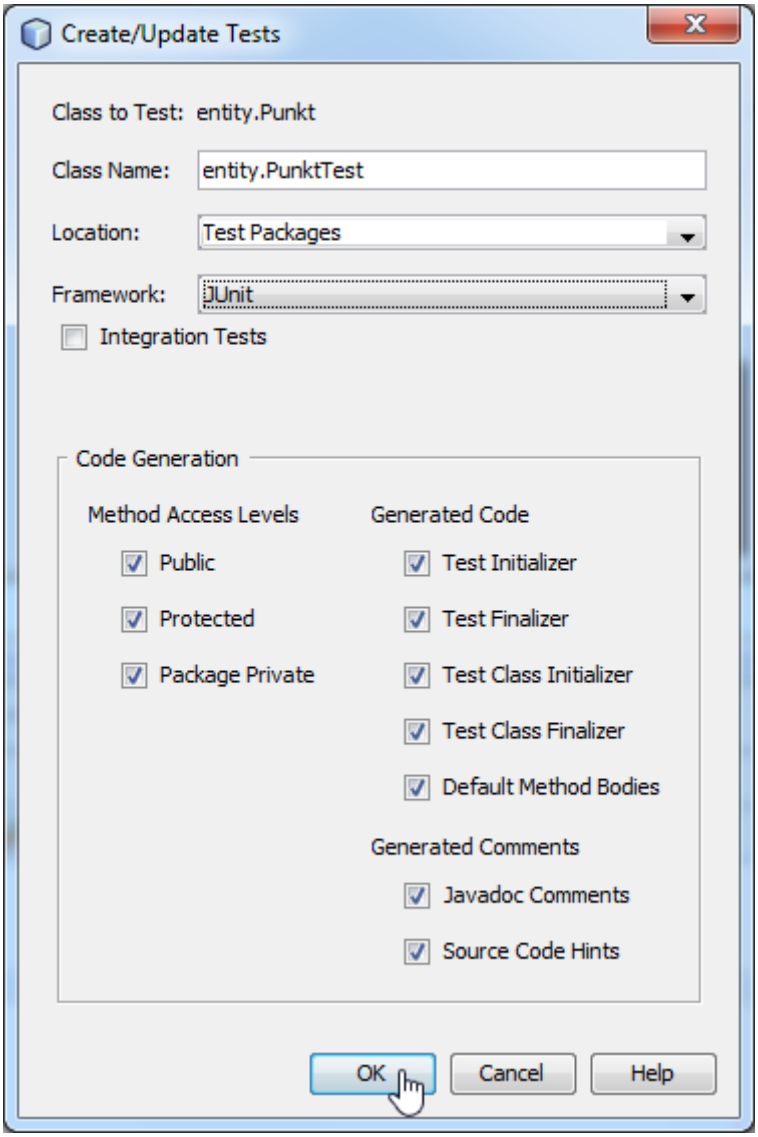
    public void setY(double y) {
        if (y < 0){
            throw new IllegalArgumentException(" y nicht negativ: " + y);
        }
        this.y = y;
    }
}
```



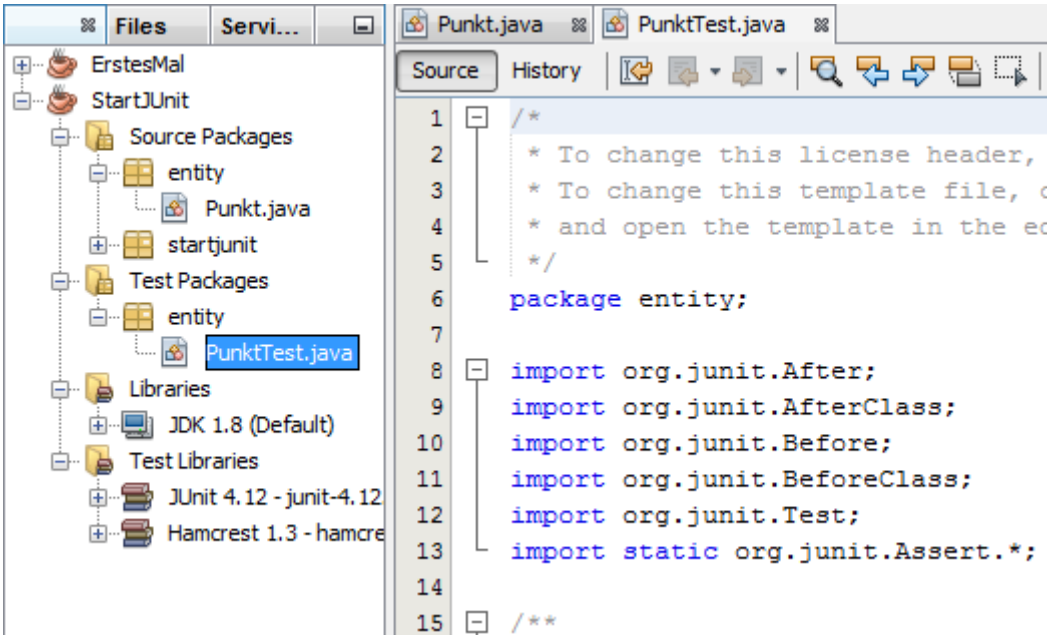
Nach einem Rechtklick auf der zu testenden Klasse wird „Tools“ und dann „Create/Update Tests“ ausgewählt.



Es ist möglich einige Programmfragmente generieren zu können, die abhängig von der zu testenden Klasse mehr oder minder sinnvoll sind. Da der Code später gelöscht werden kann, wird hier einfach „OK“ geklickt.



Es entsteht eine Klasse PunktTest, die sich im Projektunterordner „Test Packages“ befindet. Die Klasse wird in einem Paket mit dem gleichen Namen wie den der Ursprungsklasse, hier „entity“ eingeordnet“. Für die Ausführung der Tests ist wichtig, dass es sich um dasselbe Paket handelt, die Klasse Punkt zur Nutzung damit nicht importiert werden muss.
Im Projektordner „Test Libraries“ wird die aktuelle Variante von JUnit mit zusätzlich benötigen Jar-Dateien automatisch eingebunden.



Der nachfolgend gezeigte generierte Code ist meist wenig hilfreich und suggeriert, dass pro Methode nur ein Test geschrieben werden muss, was für eine professionelle Testerstellung praktisch nie gilt.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package entity;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import static org.junit.Assert.*;

/**
 *
 * @author X
 */
public class PunktTest {

    public PunktTest() {
    }

    @BeforeClass
    public static void setUpClass() {
    }
```

```
@AfterClass
public static void tearDownClass() {
}

@Before
public void setUp() {
}

@After
public void tearDown() {
}

/**
 * Test of getX method, of class Punkt.
 */
@Test
public void testGetX() {
    System.out.println("getX");
    Punkt instance = new Punkt();
    double expResult = 0.0;
    double result = instance.getX();
    assertEquals(expResult, result, 0.0);
    // TODO review the generated test code and remove the default call to fail.
    fail("The test case is a prototype.");
}

/**
 * Test of setX method, of class Punkt.
 */
@Test
public void testSetX() {
    System.out.println("setX");
    double x = 0.0;
    Punkt instance = new Punkt();
    instance.setX(x);
    // TODO review the generated test code and remove the default call to fail.
    fail("The test case is a prototype.");
}

/**
 * Test of getY method, of class Punkt.
 */
@Test
public void testGetY() {
    System.out.println("getY");
    Punkt instance = new Punkt();
    double expResult = 0.0;
    double result = instance.getY();
    assertEquals(expResult, result, 0.0);
```

```
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

    /**
     * Test of setY method, of class Punkt.
     */
    @Test
    public void testSetY() {
        System.out.println("setY");
        double y = 0.0;
        Punkt instance = new Punkt();
        instance.setY(y);
        // TODO review the generated test code and remove the default call to fail.
        fail("The test case is a prototype.");
    }

}
```

Nachdem die Tests geschrieben sind, gibt es verschiedene Wege diese auszuführen. Nach einem Rechtsklick auf der Testklasse wird „Test File“ aufgerufen.

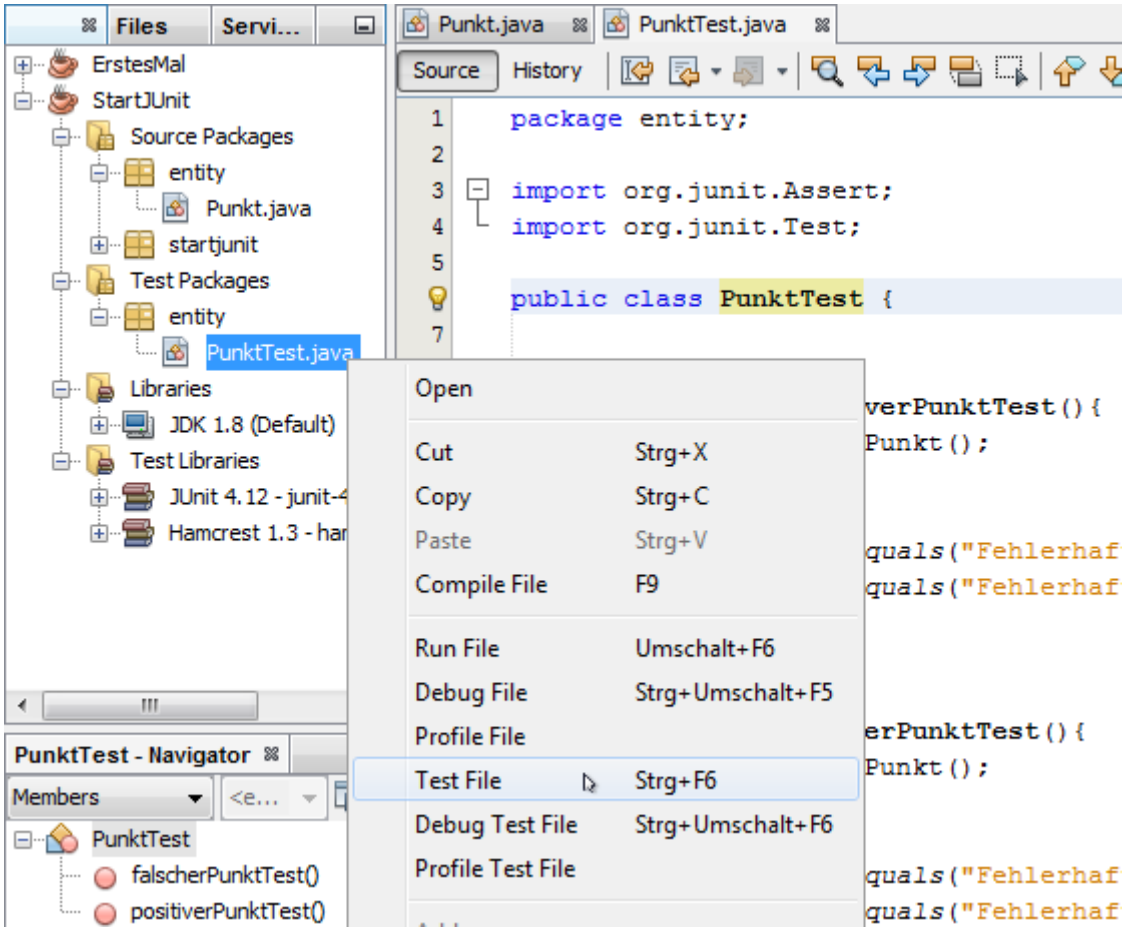
```
package entity;

import org.junit.Assert;
import org.junit.Test;

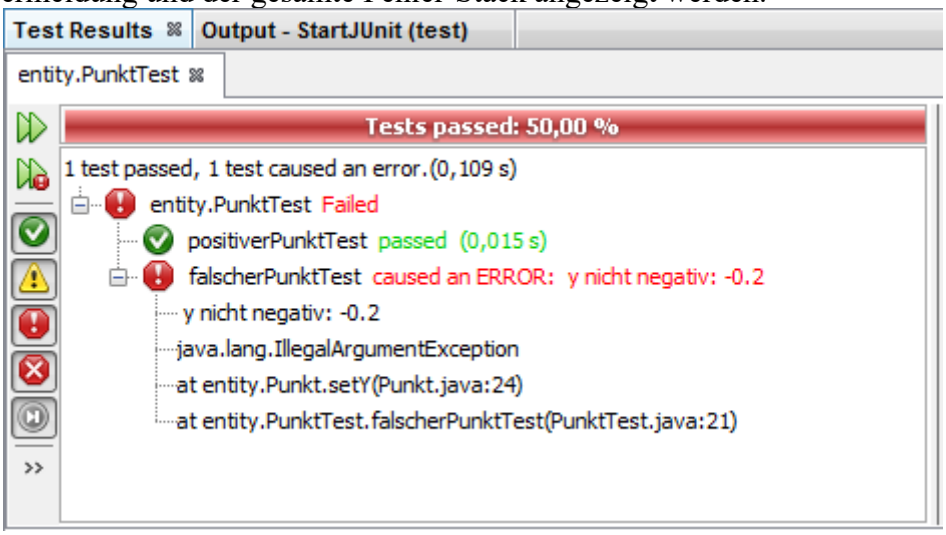
public class PunktTest {

    @Test
    public void positiverPunktTest(){
        Punkt p = new Punkt();
        p.setX(0.1);
        p.setY(0.2);
        Assert.assertEquals("Fehlerhafter x-Wert", 0.1,  p.getX(), 1e-12);
        Assert.assertEquals("Fehlerhafter y-Wert", 0.2,  p.getY(), 1e-12);
    }

    @Test
    public void falscherPunktTest(){
        Punkt p = new Punkt();
        p.setX(0.1);
        p.setY(-0.2);
        Assert.assertEquals("Fehlerhafter x-Wert", 0.1,  p.getX(), 1e-12);
        Assert.assertEquals("Fehlerhafter y-Wert", -0.2,  p.getY(), 1e-12);
    }
}
```

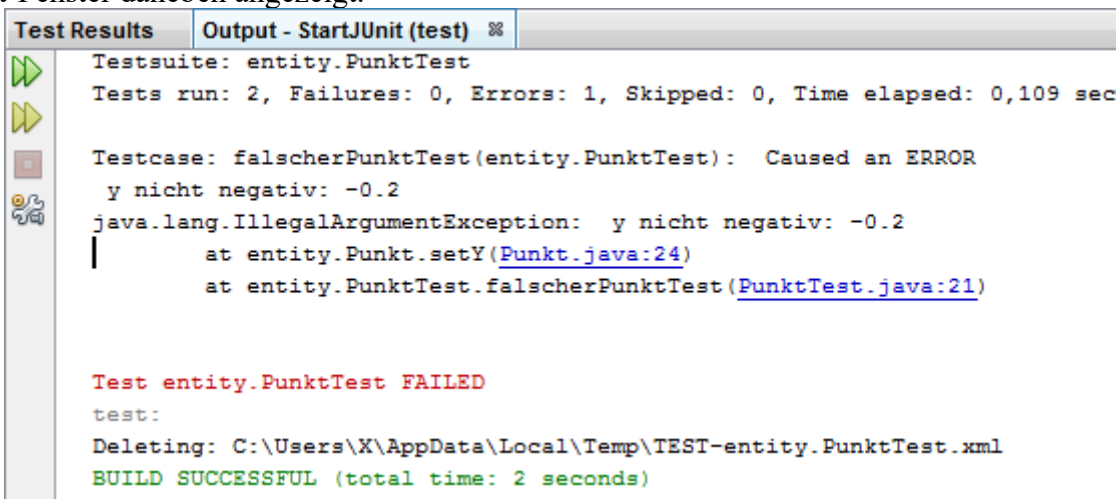


Die Ergebnisse werden unten im Output-Bereich in einem eigenen Reiter „Test Results“ ausgegeben. Fehlerhafte Tests können aufgeklappt werden und enthalten mehr Informationen, wobei oft nicht die gesamte Fehlermeldung und der gesamte Fehler-Stack angezeigt werden.

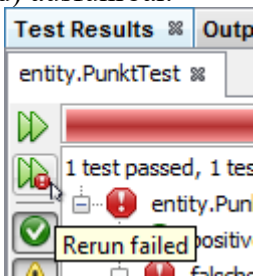


Sollten die Tests Ausgaben enthalten oder es zu Fehlern kommen, werden diese Meldungen im

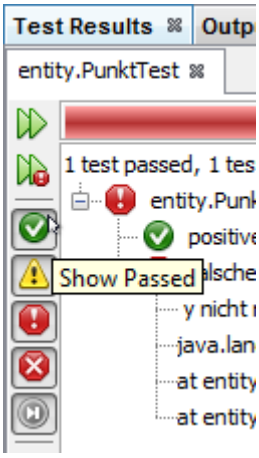
Output-Fenster daneben angezeigt.



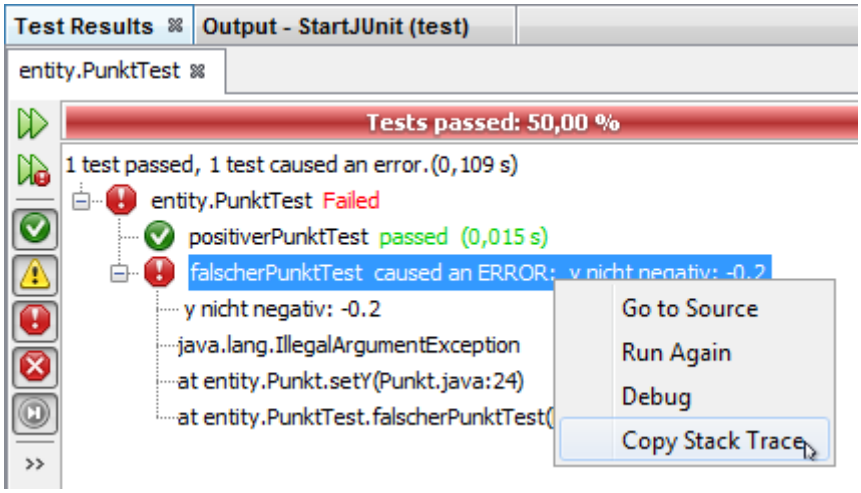
Über die oberen beiden Knöpfe am linken Rand der Test-Ausgabe sind alle Tests erneut (Rerun) oder alle fehlgeschlagenen Tests (Rerun failed) ausführbar.



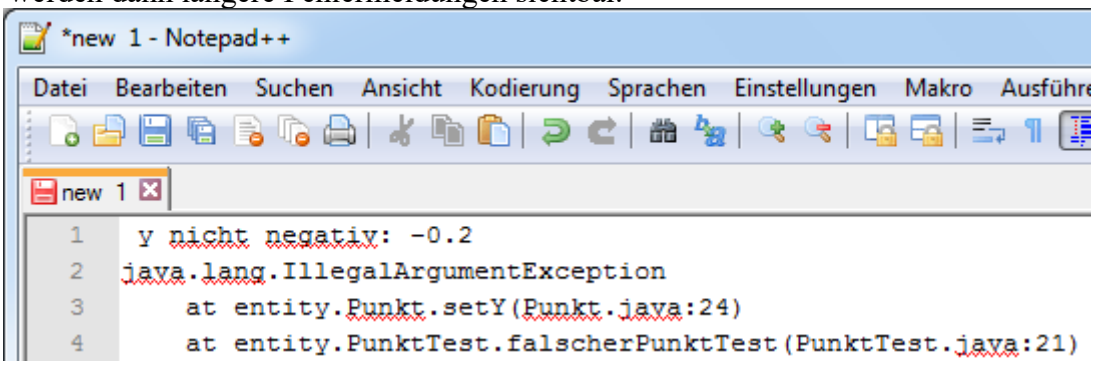
Mit den Knöpfen darunter können bestimmte Testergebnisse ein- bzw. ausgeblendet werden, wenn z. B. nur gescheiterte Tests sichtbar sein sollen.



Wird eine detailliertere Fehlerinformation benötigt, wird ein Rechtsklick auf den gescheiterten Testfalls gemacht und „Copy Stack Trace“ ausgewählt. Der Text befindet sich im lokalen Speicher und kann dann in ein beliebiges Textwerkzeug hineinkopiert („Paste“, Strg-V) werden.

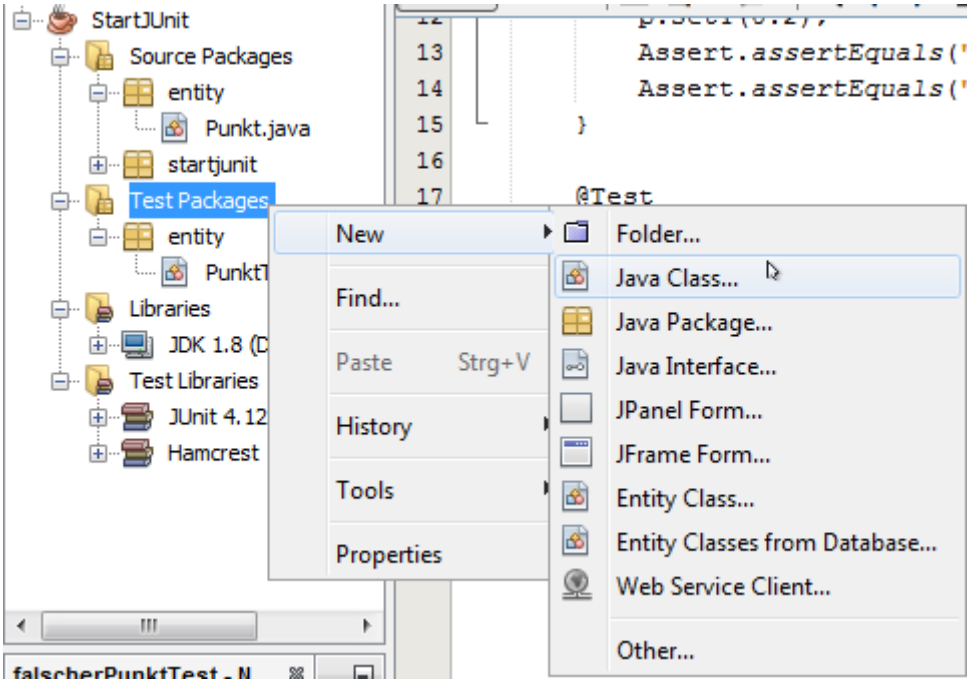


Hierbei werden dann längere Fehlermeldungen sichtbar.

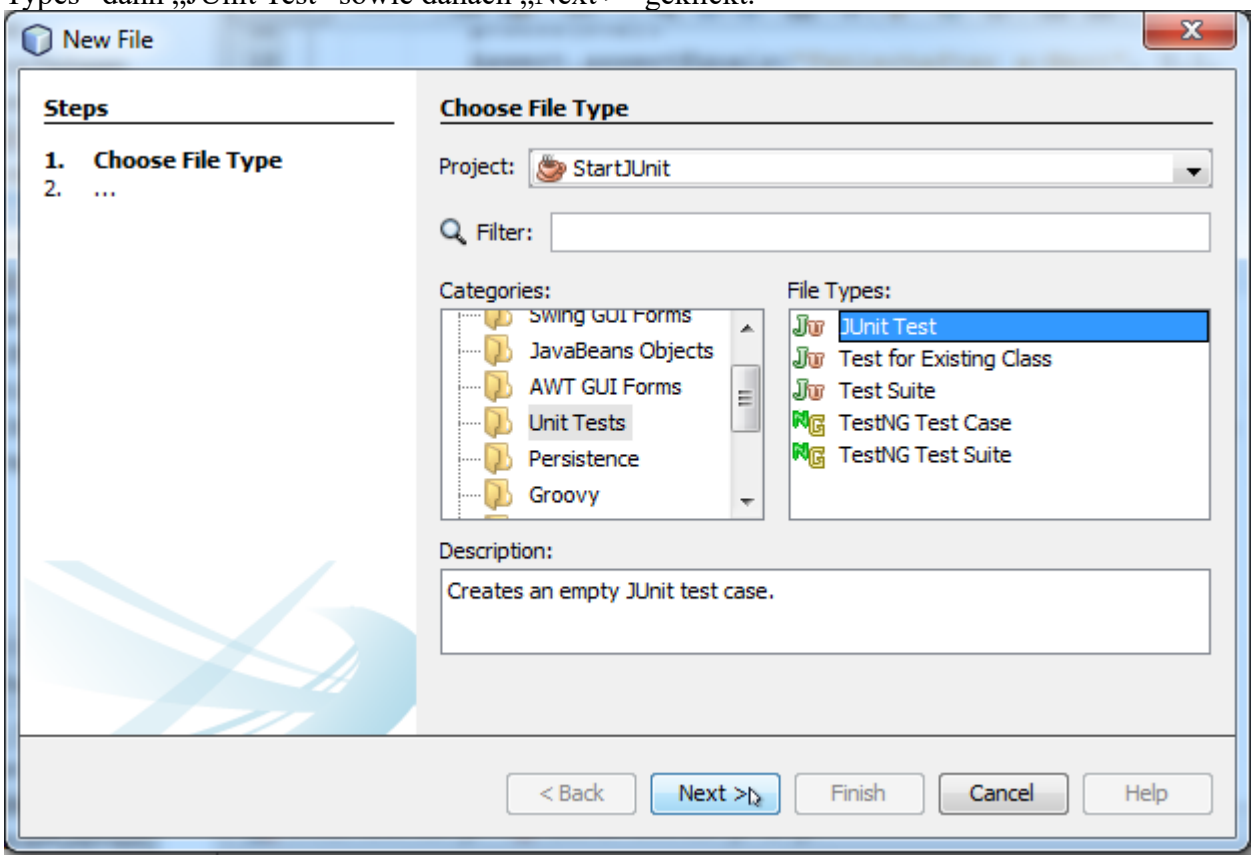


5.2 Erstellung einer neuen Test-Klasse

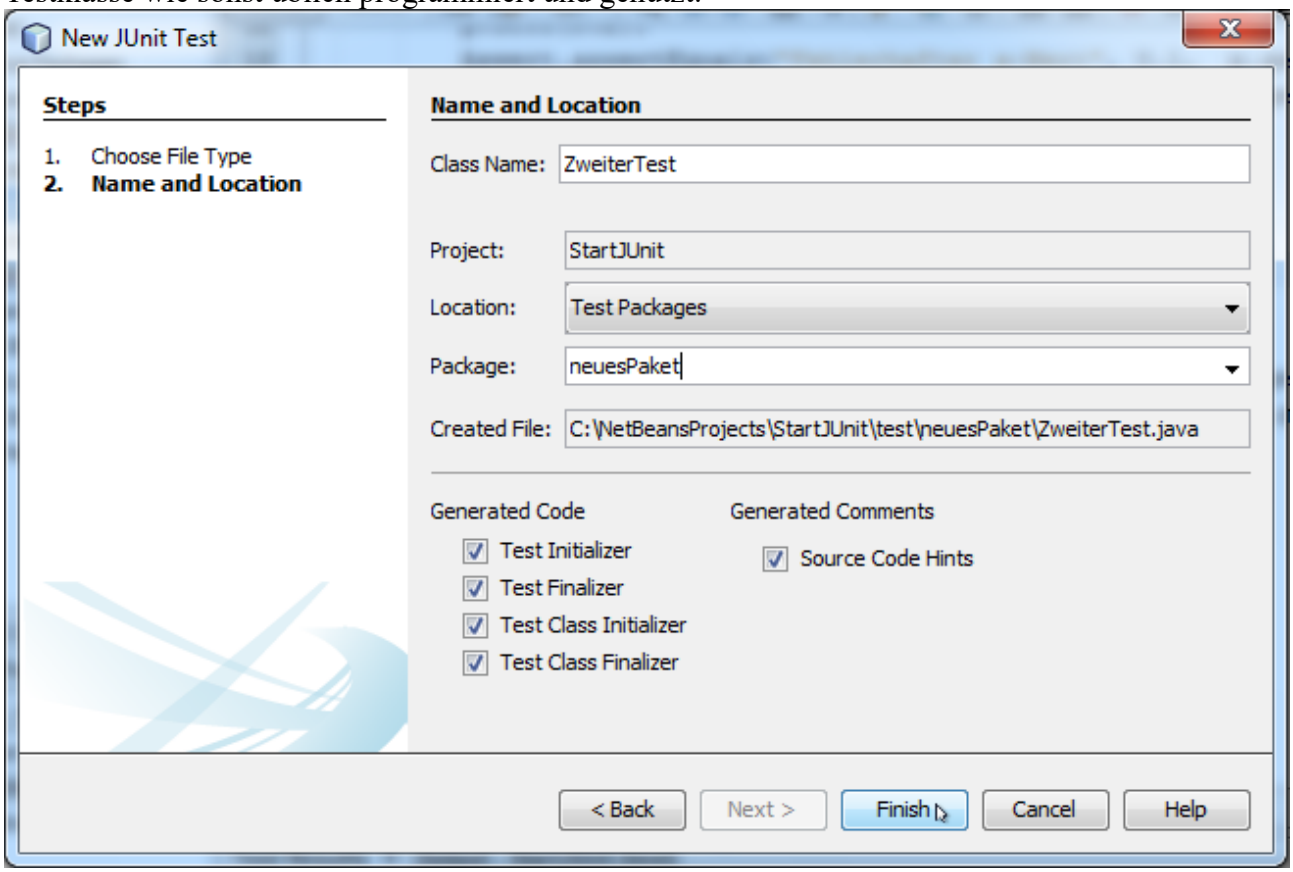
Neue Tests sind unter „Test Packages“ zu erstellen. Dies kann als normale Java Klasse oder als „JUnit Test“ erfolgen, wodurch in der Testklasse bereits einige Code-Fragmente erzeugt werden.



Sollten sich „JUnit Test“ nicht direkt unter „New“ befinden, wird auf „Other…“ in der vorherigen Abbildung geklickt. Unter „Categories“ wird „Unit Tests“ und dann auf der rechten Seite unter „File Types“ dann „JUnit Test“ sowie danach „Next >“ geklickt.

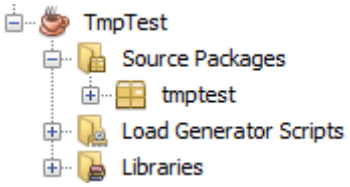


Hier ist wie bei der normalen Klassenerzeug der Klassenname (Class Name) und das Paket (Package) anzugeben. Weiterhin sind einige Auswahlen für den zu generierenden Code möglich, auf die allerdings verzichtet werden kann. Die Erstellung wird über „Finish“ abgeschlossen. Danach wird die Testklasse wie sonst üblich programmiert und genutzt.

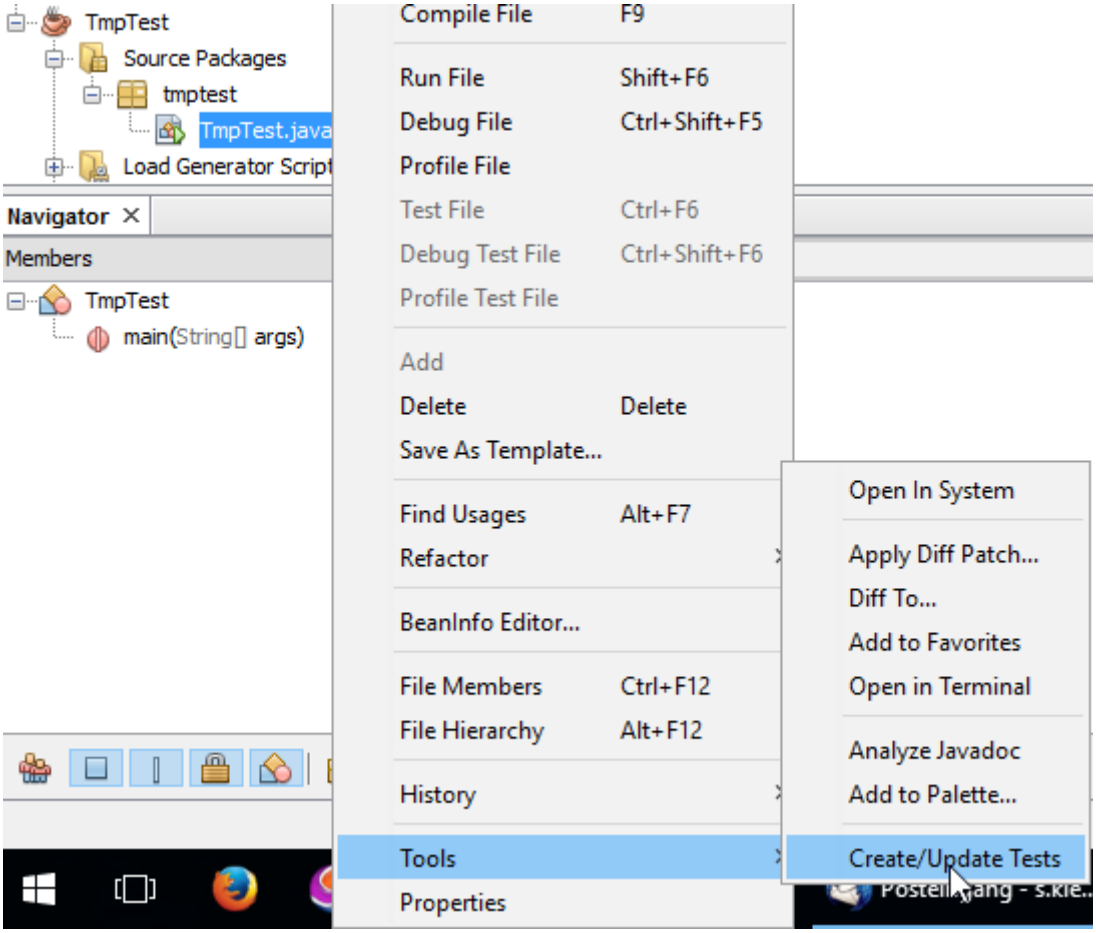


5.3 Einblendung von „Test Packages“

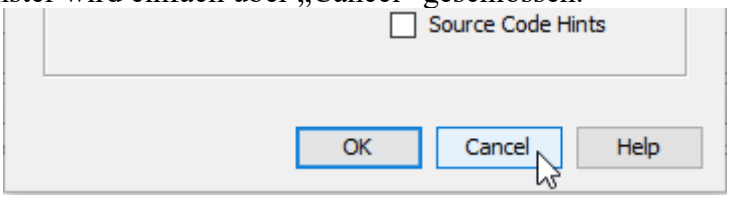
Abhängig von der NetBeans-Version kann es sein, dass für Java-Projekte nicht direkt der Ordner „Test Packages“ angezeigt wird.



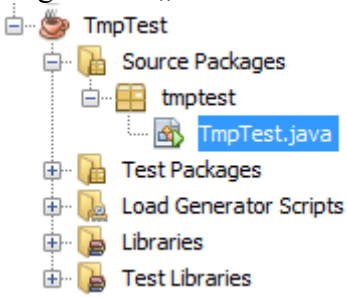
Soll nur der Ordner hinzukommen, wird nur der erste Schritt einer Testerstellung über NetBeans ausgeführt und dann dieser Schritt abgebrochen. Dazu kann z. B. mit einem Rechtsklick auf einer Klasse „Tools“ und dann „Create/Update Tests“ ausgewählt werden.



Das sich öffnende Fenster wird einfach über „Cancel“ geschlossen.

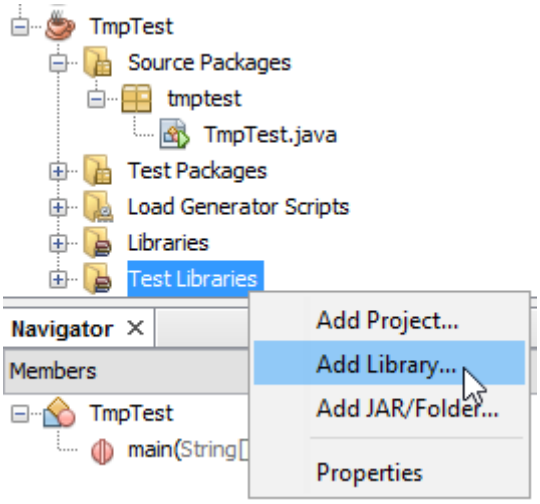


Danach stehen die Ordner „Test Packages“ und „Test Libraries“ zur Verfügung.

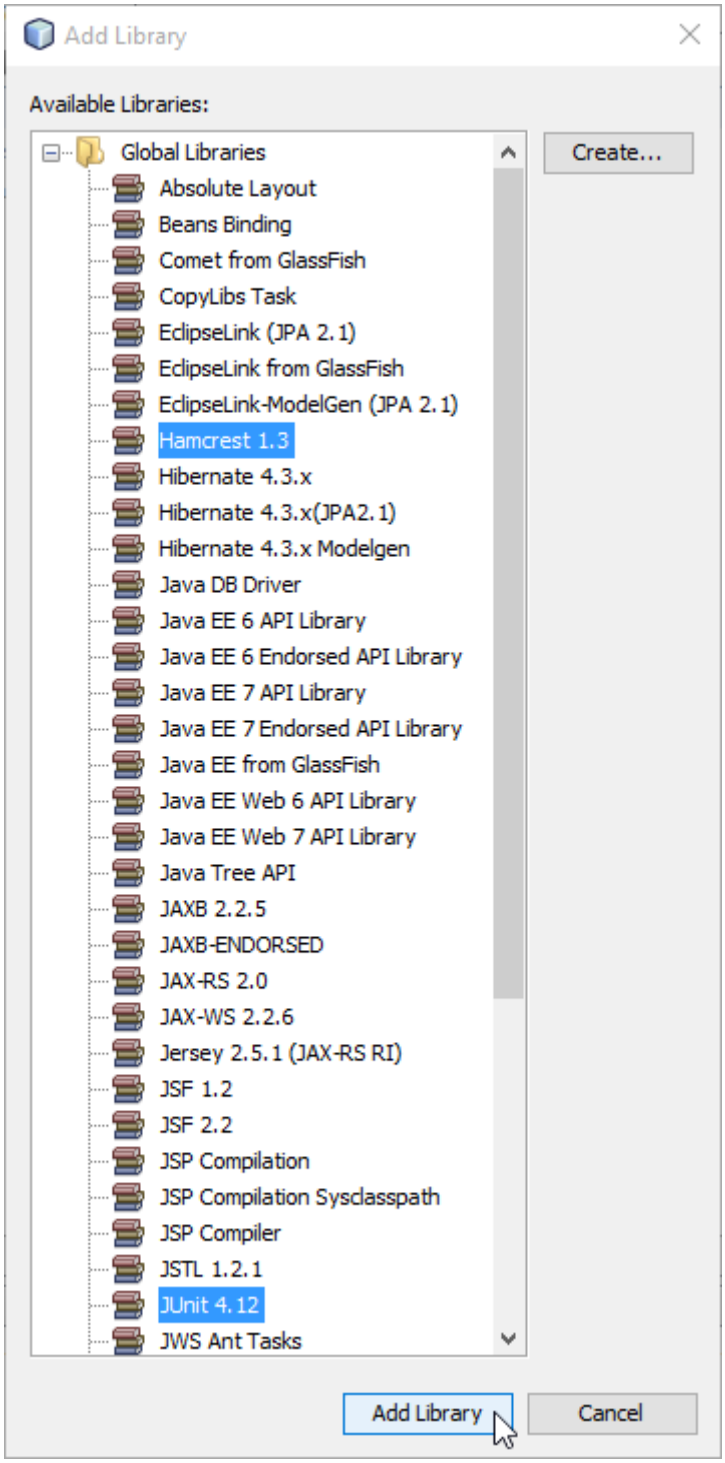


5.4 Direkte Nutzung der JUnit-Library von NetBeans

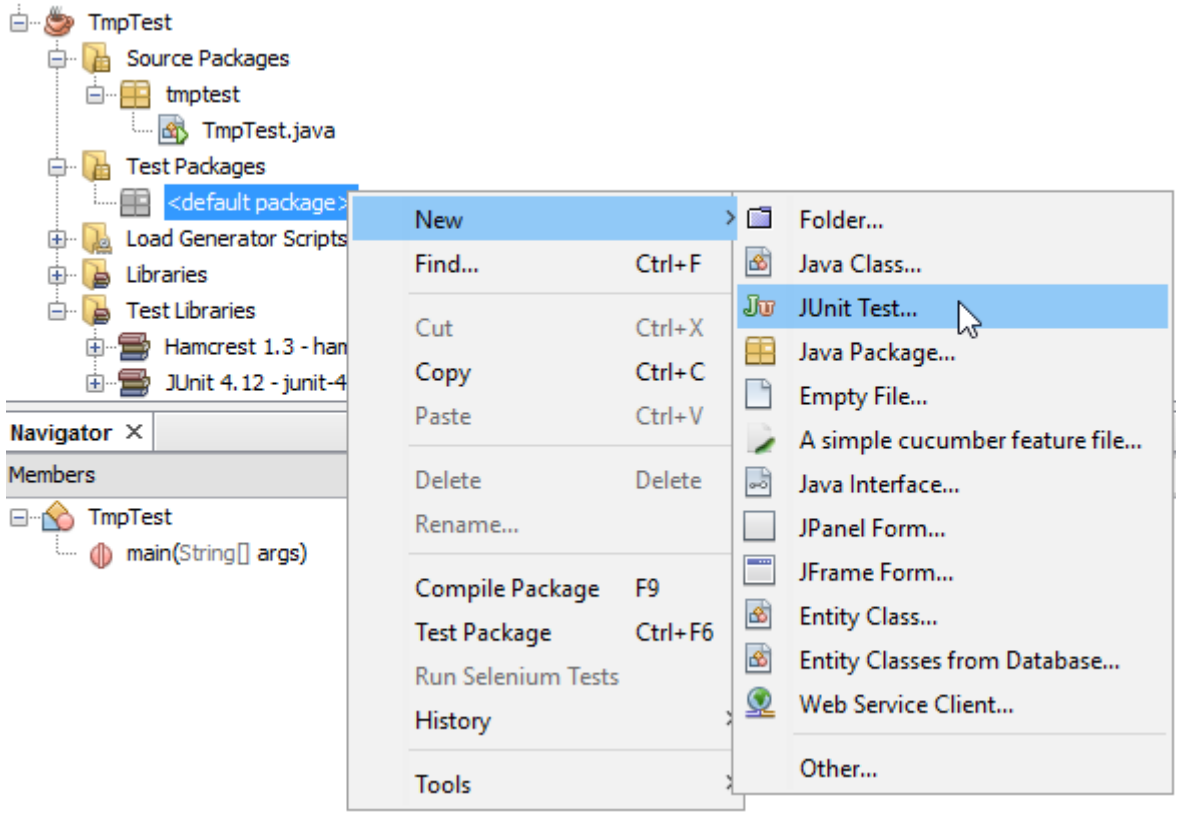
Soll JUnit einfach ohne Code-Generierung genutzt werden, ist JUnit in das Projekt einzubinden. Dazu kann auf die von NetBeans verwalteten Jar-Dateien über eine Library zugegriffen werden. Zu diesem Zweck wird ein Rechtsklick auf dem Ordner „Test Libraries“ gemacht und „Add Library…“ ausgewählt.



Hier werden mit gedrückter Strg-Taste Hamcrest und JUnit ausgewählt und über „Add Library“ hinzugefügt.

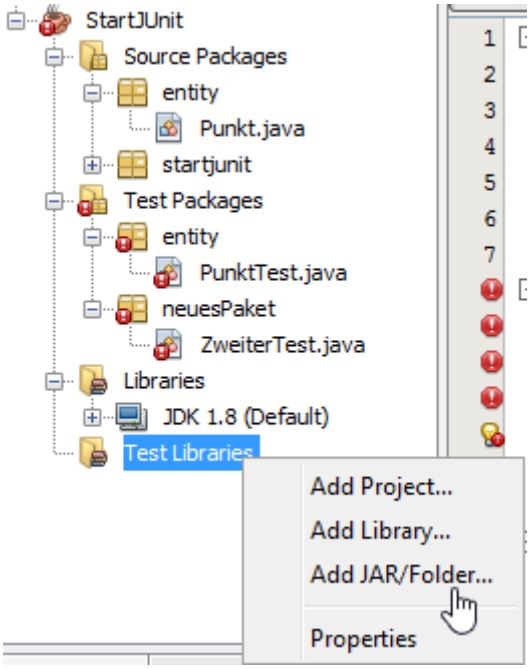


Nun können unter „Test Packages“ die Tests entwickelt werden. Wieder gilt, dass ein identischer Paketname dafür sorgt, dass die Testklasse sich in diesem Paket befindet und so die zugehörigen Rechte hat. Die Testausführung verläuft identisch zu den in „5.1 Vollständige Nutzung der JUnit-NetBeans-Unterstützung“ beschriebenen Schritten.

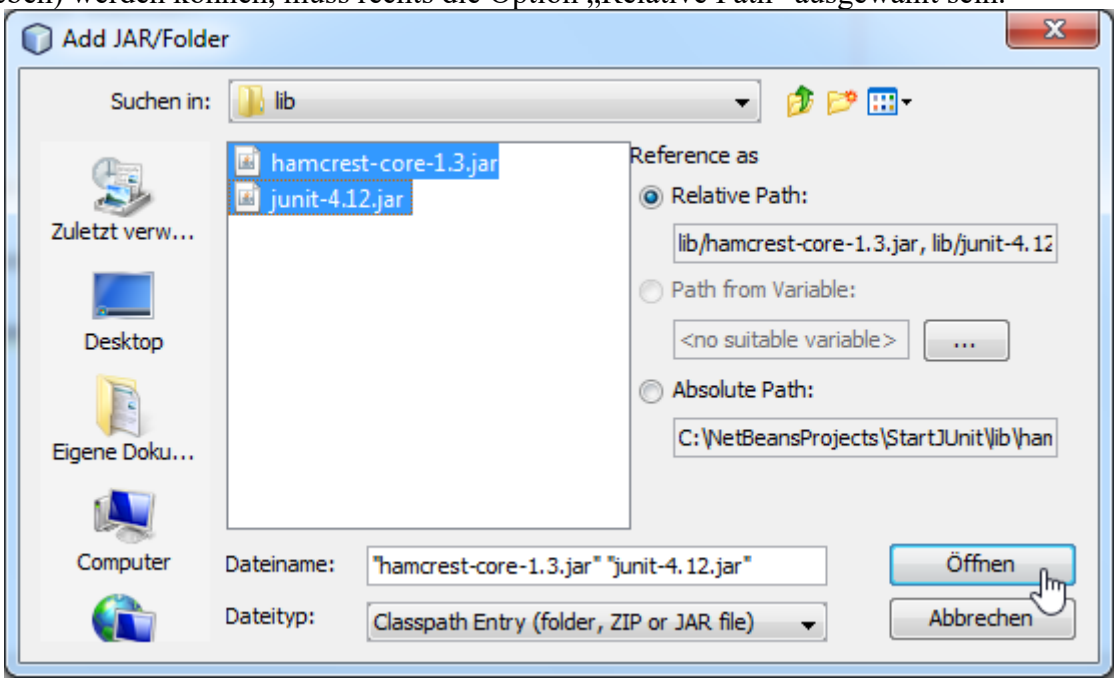


5.5 Nutzung eigener Test-Bibliotheken

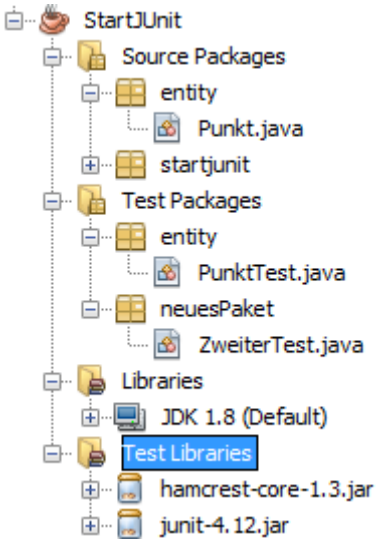
Soll nicht die JUnit-Bibliothek von NetBeans oder sollen weitere Test-Bibliotheken genutzt werden, sind diese auch direkt recht einfach nutzbar. Der Ansatz ist fast identisch zur Nutzung von Bibliotheken, wie er in „4.6 Nutzung von Bibliotheken/Jar-Files (NetBeans 8)“ beschrieben ist. Der einzige Unterschied ist, dass die Bibliotheken unter „Test Libraries“ hinzugefügt werden. Dazu wird ein Rechtsklick auf „Test Libraries“ gemacht und „Add JAR/Folder …“ angeklickt.



Es wird zu dem Ordner gesteuert, in den vorher die gewünschten Libraries kopiert wurden, diese mit gedrückter Strg-Taste markiert und mit „Öffnen“ hinzugefügt. Damit die Projekte kopiert (und abgegeben) werden können, muss rechts die Option „Relative Path“ ausgewählt sein.



Die Bibliotheken werden unter „Test Libraries“ angezeigt und können im Test-Code genutzt werden. Theoretisch ist auch der Ordner „Libraries“ nutzbar, da in Tests auch auf diese Libraries zugegriffen werden kann, allerdings sorgt die Trennung der Libraries dafür, dass die nur zum Testen benötigten Libraries nicht mit in das auszuliefernde Produkt integriert werden.

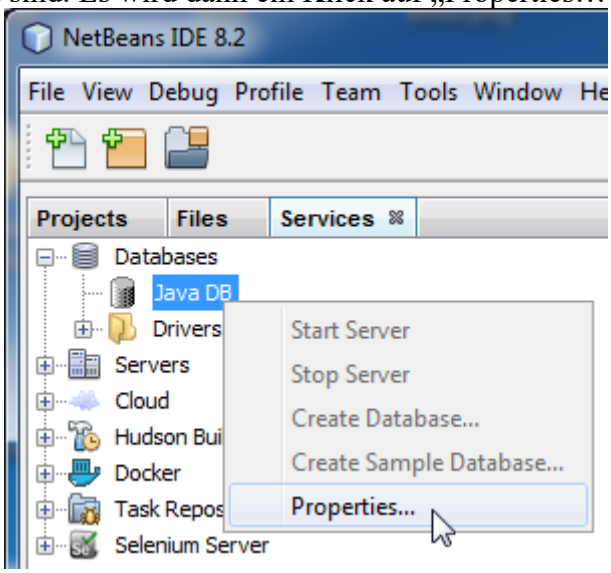

StartJUnit
Source Packages
entity
Punkt.java
startjunit
Test Packages
entity
PunktTest.java
neuesPaket
ZweiterTest.java
Libraries
JDK 1.8 (Default)
Test Libraries
hamcrest-core-1.3.jar
junit-4.12.jar

6 NetBeans und Datenbanken (NetBeans 8)

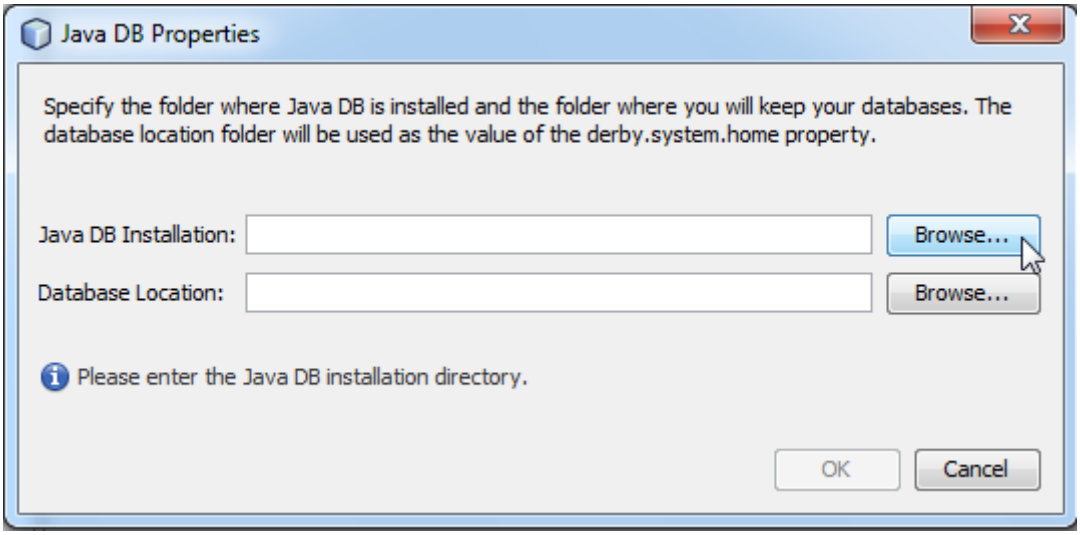
6.1 Integration von Apache Derby /Java DB in NetBeans

Java DB geht auf die Datenbank Apache Derby (https://db.apache.org/derby/) zurück, bei der es sich um eine vollwertige Datenbank mit Transaktionssteuerung und der Unterstützung fast aller SQL-Befehle handelt. NetBeans ermöglicht die Erstellung und Bearbeitung von Datenbanken mit einem recht mächtigen Verwaltungssystem. Die Datenbanken werden typischerweise im Zusammenhang mit Web-Applikationen genutzt, stehen aber auch sonst zur Verfügung.

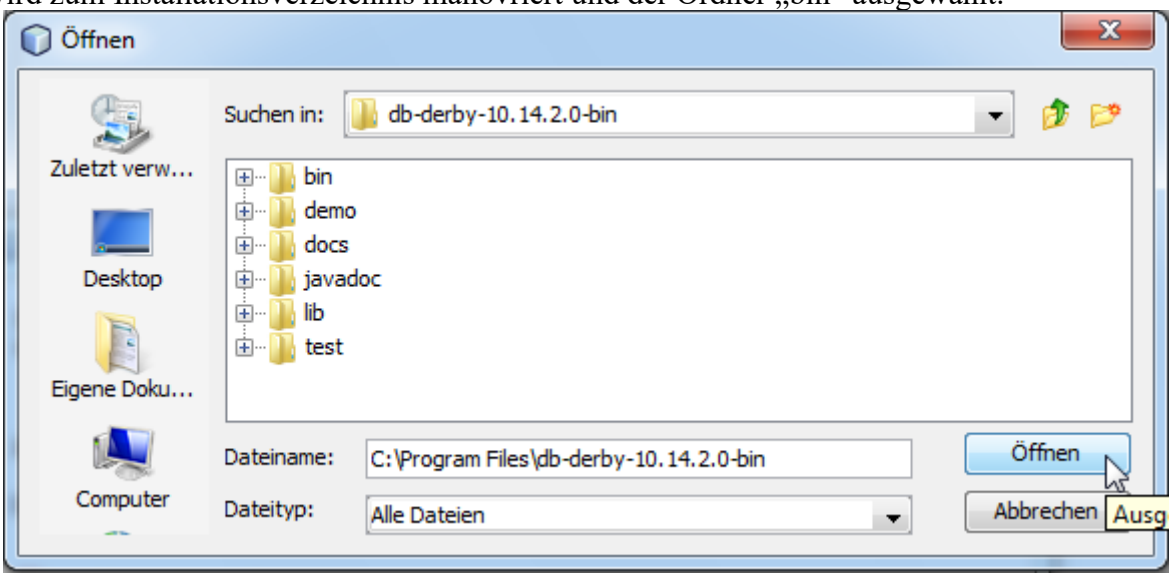
Wird eine JDK Version vor Java 7 oder nach JDK 8u172 genutzt, ist Java DB, bzw. Apache Derby, nicht direkt mit Java installiert worden, so dass die vorher installierte Datenbank, siehe „1.3 Installation von Apache Derby (früher Java DB“, in NetBeans bekannt gemacht werden muss. Die fehlende Datenbank wird deutlich, wenn unter dem Reiter „Services“, der Punkt „Databases“ erweitert und ein Rechtsklick auf „JavaDB“ gemacht wird und dann alle Möglichkeiten bis auf „Properties…“ ausgegraut sind. Es wird dann ein Klick auf „Properties…“ gemacht.



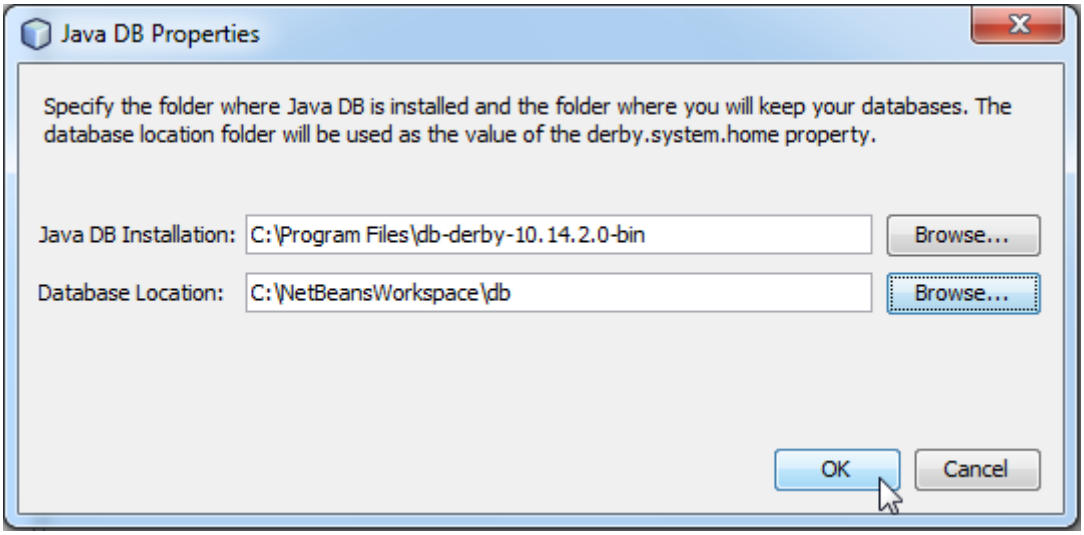
Nun wird der obere Eintrag „Java DB Installation“ gesetzt. Dazu wird ein Klick auf „Browse…“ gemacht.



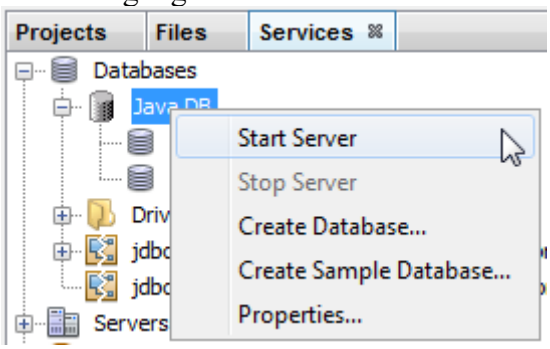
Es wird zum Installationsverzeichnis manövriert und der Ordner „bin“ ausgewählt.



Im Eintrag unter „Database Location“ kann der Speicherort für Datenbanken fesrtgelegt werden. Hier gibt es zuminmdest zwei Ansätze, es gibt ein zentrales Verzeichnis in dem alle Datenbanken liegen, was den Zugriff in NetBeans sehr einfach macht, oder es wird in jedem Projekt ein Unterordsner „db“ angelegt, in dem die zum Projekt gehörende Datenbank installiert wird. Dies macht die Projektzuordnung wesentlich deutlicher, lässt aber der Nutzung mehrerer Datenbanken nicht zu. Bearbeitet man dann mehrere Projekte gleichzeitig, muss der Pfad zu den Datenbanken jeweils eingestellt werden.
Die Integration wird mit „OK“ abgeschlossen.

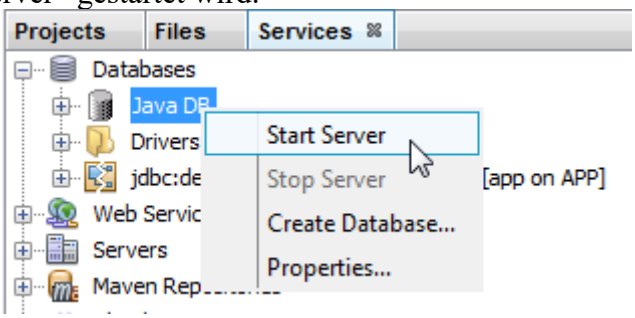


Danach stehen die benötigten Funktionalitäten zum Starten des Datenbank-Servers und zum Anlegen neuer Datenbanken zur Verfügung.



6.2 Anlegen einer neuen Datenbank

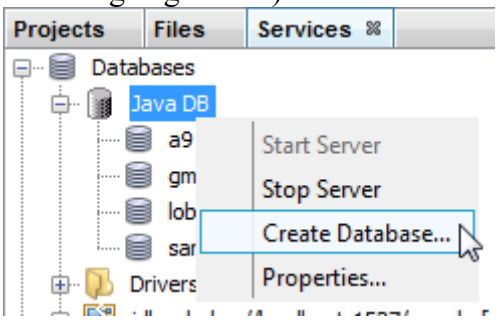
Es soll jetzt eine neue Datenbank eingerichtet werden, die beispielhaft von JDBC aus genutzt werden soll. Die Datenbank läuft dabei auf dem gleichen Rechner, also unter der Adresse localhost. Mit einem Rechtsklick auf Java DB kann man sich über den Status des Datenbankservers informieren, der für das Beispiel mit „Start Server“ gestartet wird.



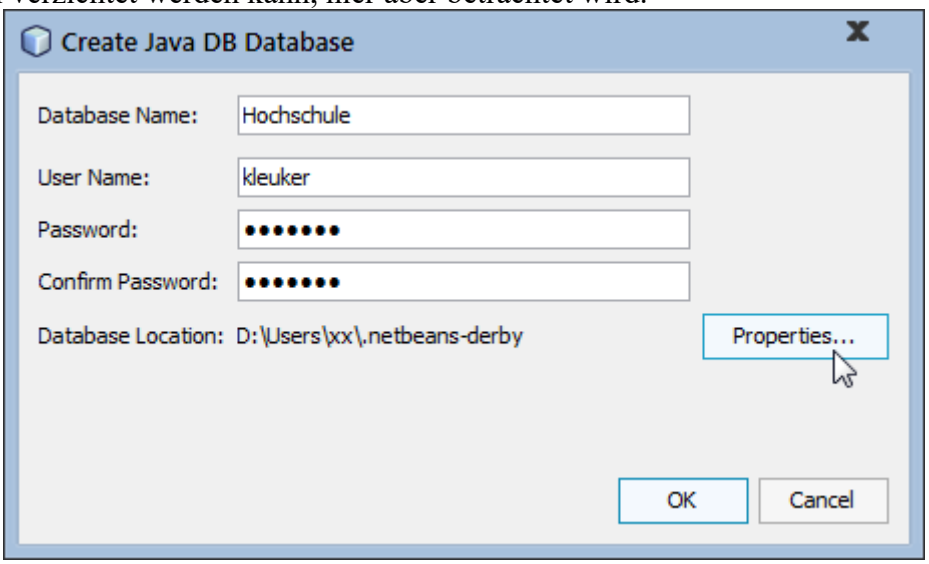
Im Output-Fenster wird ein Text der folgenden Form ausgegeben. Die Port-Nummer ist relevant, wenn man eine Verbindung mit der Datenbank aufnehmen möchte.

```
Mon Sep 15 18:30:35 CEST 2014 : Sicherheitsmanager mit einfacher Server-
Sicherheitsrichtlinie installiert.
Mon Sep 15 18:30:35 CEST 2014 : Apache Derby Network Server 10.8.3.2 - (1557835) wurde
gestartet und ist bereit, Verbindungen am Port 1527 zu akzeptieren.
```

Danach wird im gleichen Menü „Create Database...“ gewählt (was auch ohne Server-Start möglich wäre, der Server wird dann beim Erzeugen gestartet).



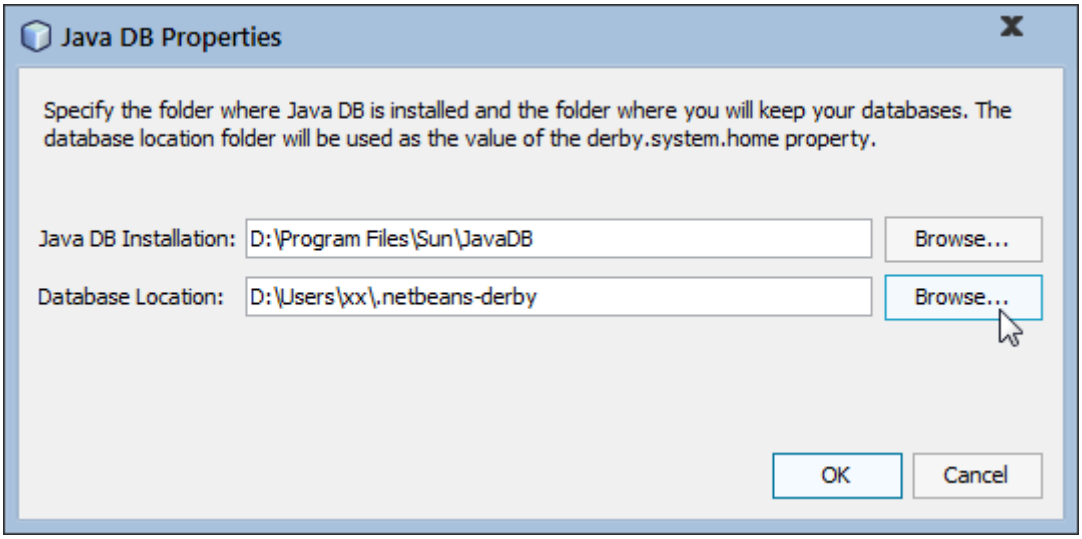
Es werden die vorgegebenen Felder gefüllt, dabei kann auf einen Nutzer und ein Passwort verzichtet werden, wovon aber abgeraten wird. Für weitere Einstellungen wird auf „Properties“ geklickt, worauf im Normalfall verzichtet werden kann, hier aber betrachtet wird.



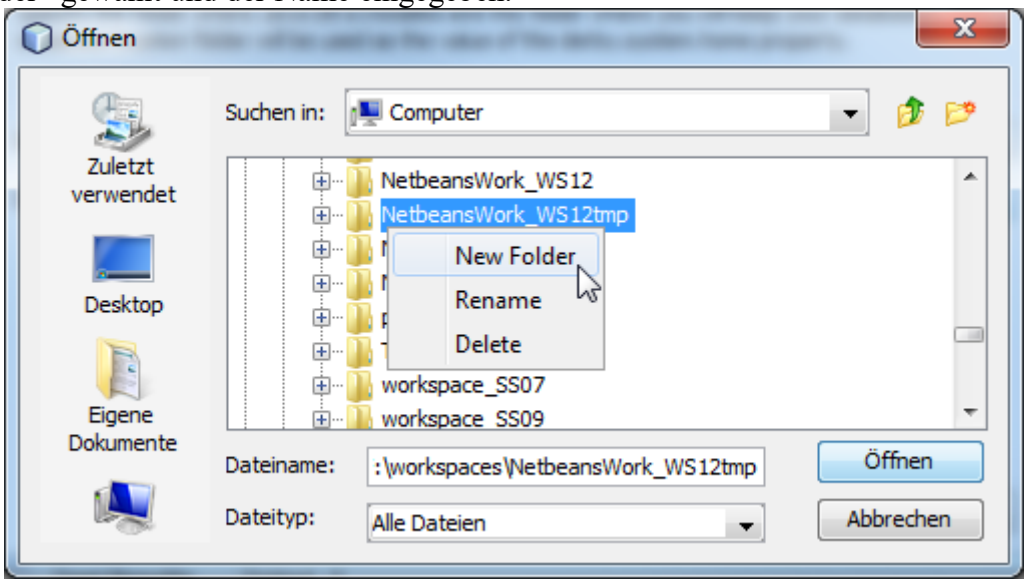
Ist die Einrichtung der Datenbank und der Pfade dem im vorherigen Unterkapitel gezeigten Ansatz gefolgt, kann auf die Nutzung von „Properties“ verzichtet und die Datenbank mit „OK“ angelegt werden.

Unter Properties sieht man, welche Datenbanksoftware genutzt und wo die Datenbank gespeichert wird. Der letzte Ort ist passend für alle Programme mit Java DB-Nutzung abzuändern. Da das Datenbanksystem dateibasiert arbeitet, kann die jeweilige Datenbank leicht verschoben und auch gesichert werden. In der Hochschule kann die vorgegebene Einstellung übernommen werden, alternativ ist die „Database Location“ auf ein Nutzerlaufwerk umzuändern, in dem man z. B. Datenbanken sammelt. Die hier vorgeschlagene Variante ist, die Datenbank über den Projektordnern abzulegen, in denen sie genutzt werden sollen. Ein direktes Anlegen der Datenbanken in den jeweiligen Projektordnern kann zu Problemen beim Kompilieren und Deployen von Web-

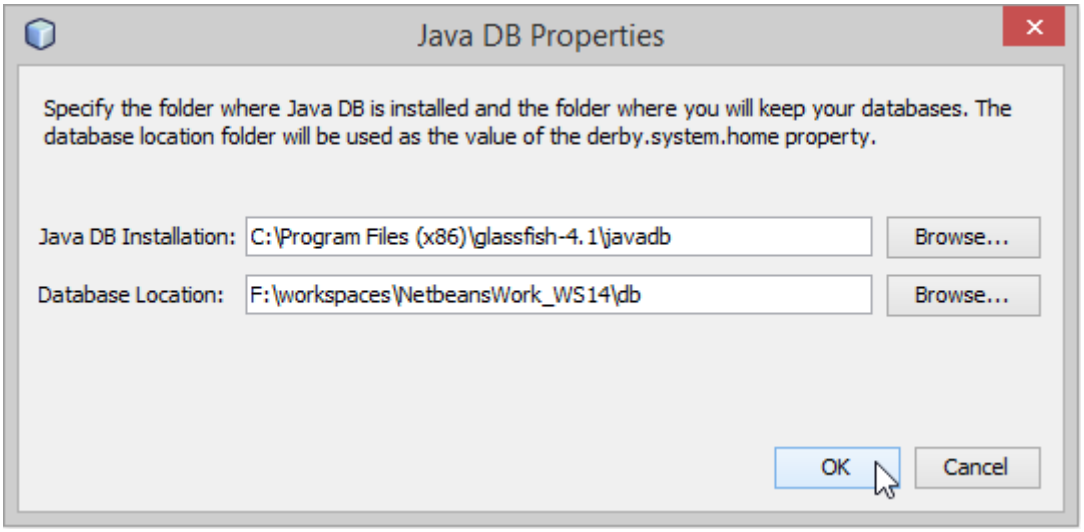
Applikationen führen.



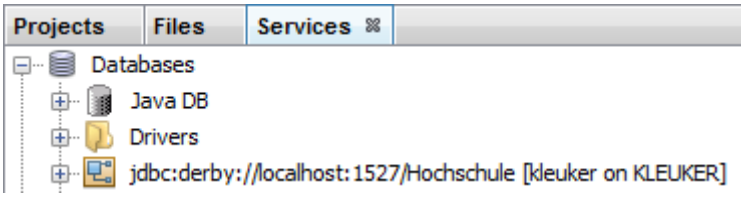
Dazu ist es sinnvoll, im Projekt zunächst einen Ordner anzulegen, was auch hier direkt möglich ist. Dazu wird ein Rechtsklick auf dem Ordner gemacht, dem ein Unterordner hinzugefügt werden soll, „New Folder“ gewählt und der Name eingegeben.



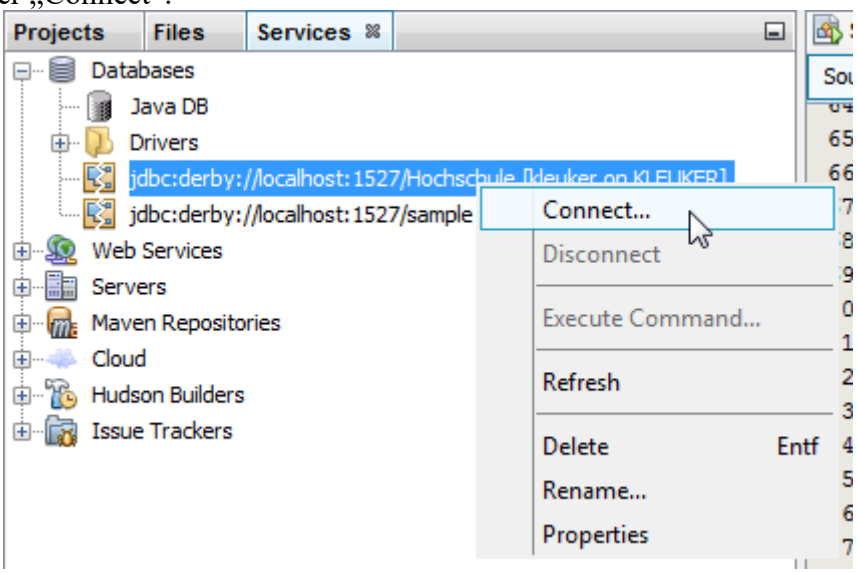
Danach wird die Erstellung mit einem „OK“ im vorherigen Fenster abgeschlossen. Die folgende Abbildung zeigt die hier genutzten Einstellungen.



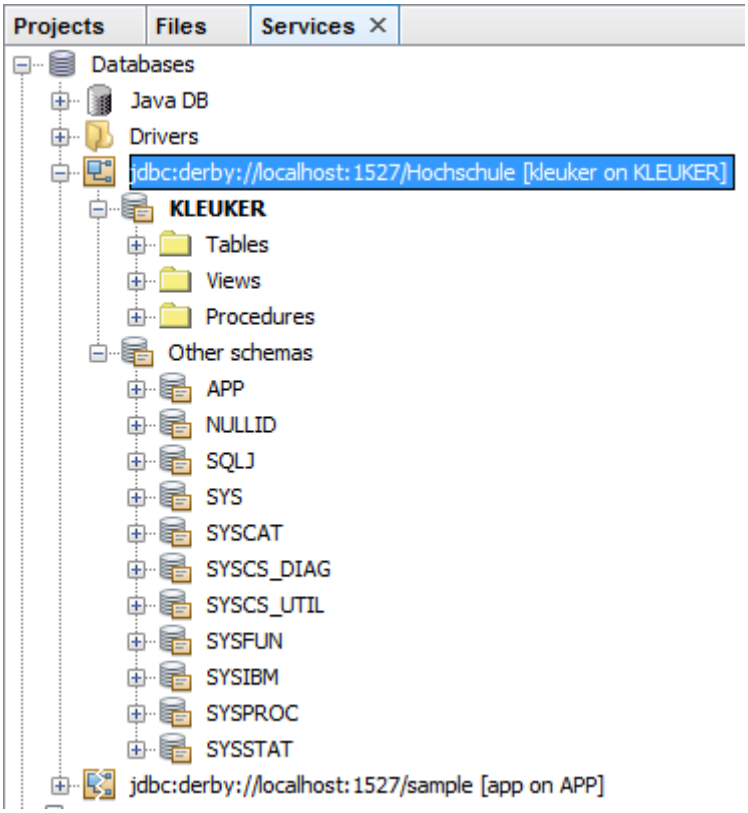
Die neue Datenbank ist auch im Browser sichtbar. Man sieht weiterhin den Nutzernamen und das zugehörige Schema (der Datenbankbereich). Implizit merkt sich das System das Passwort für den Nutzer.



Möchte man eine Tabelle anlegen, so macht man einen Rechtsklick auf der Datenbank und verbindet sich zunächst über „Connect“.

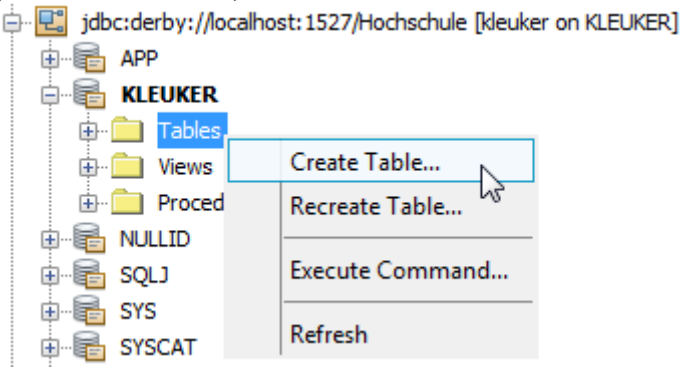


Man sieht dann eine Unterstruktur mit verschiedenen für das Datenbanksystem relevanten Bereichen.

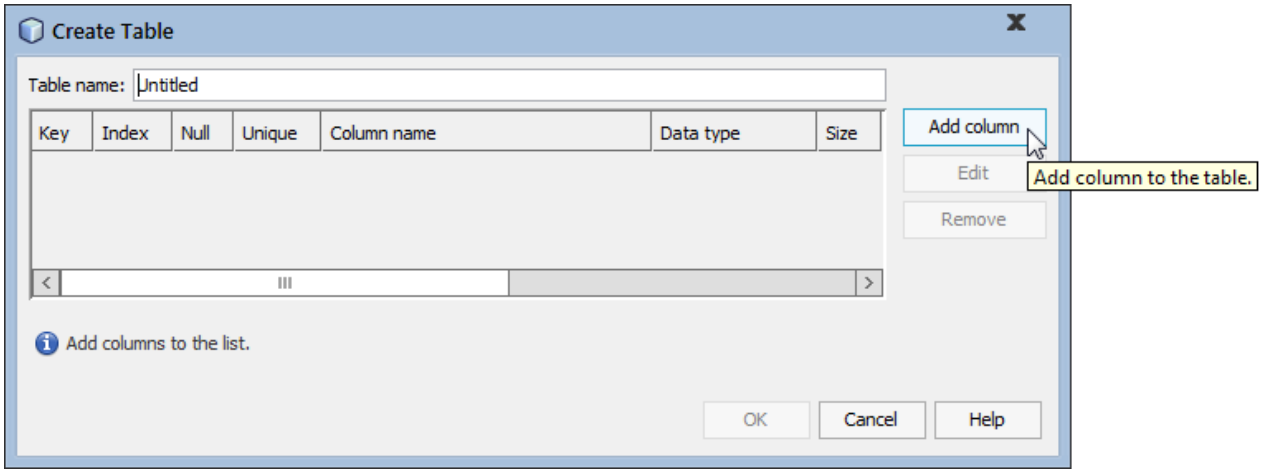


6.3 Anlegen einer Tabelle

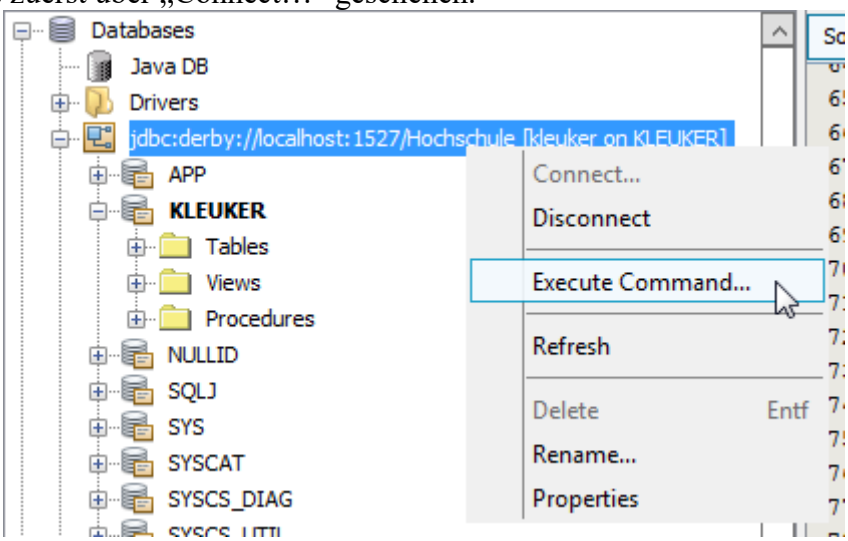
Durch einen Rechtsklick auf den Ordner Tables erhält man mit „CREATE TABLE“ die Möglichkeit, eine Tabelle anzulegen. Diese Möglichkeit wird hier nur vorgeführt, aber im Folgenden nicht genutzt, da ein SQL-Skript eingesetzt werden soll, was danach beschrieben ist.



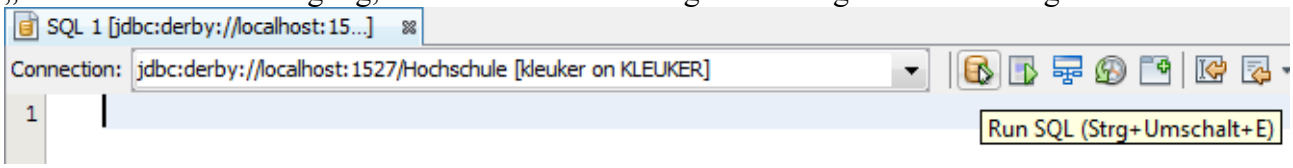
Im darauf sich öffnenden Fenster kann man dann eine Tabelle zusammenstellen. In der folgenden Beschreibung wird allerdings die direkte Befehlseingabe bevorzugt.



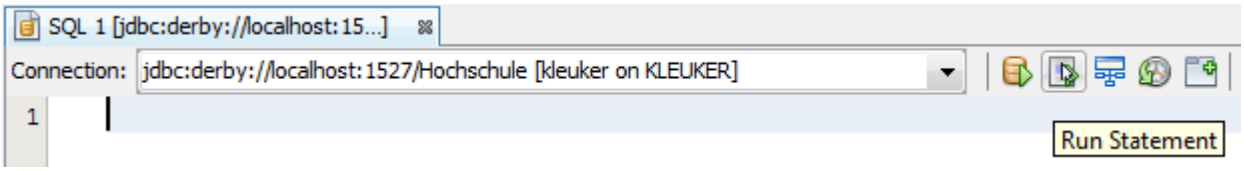
Möchte man einfach beliebige SQL-Befehle ausführen, wird bei der Auswahl unter Tables einfach „Execute Command...“ genutzt. Genau der gleiche Befehl steht auch direkt durch einen Rechtsklick auf der Datenbank zur Verfügung. Falls die Verbindung zur Datenbank-Instanz noch nicht aufgebaut wurde, muss dies zuerst über „Connect…“ geschehen.



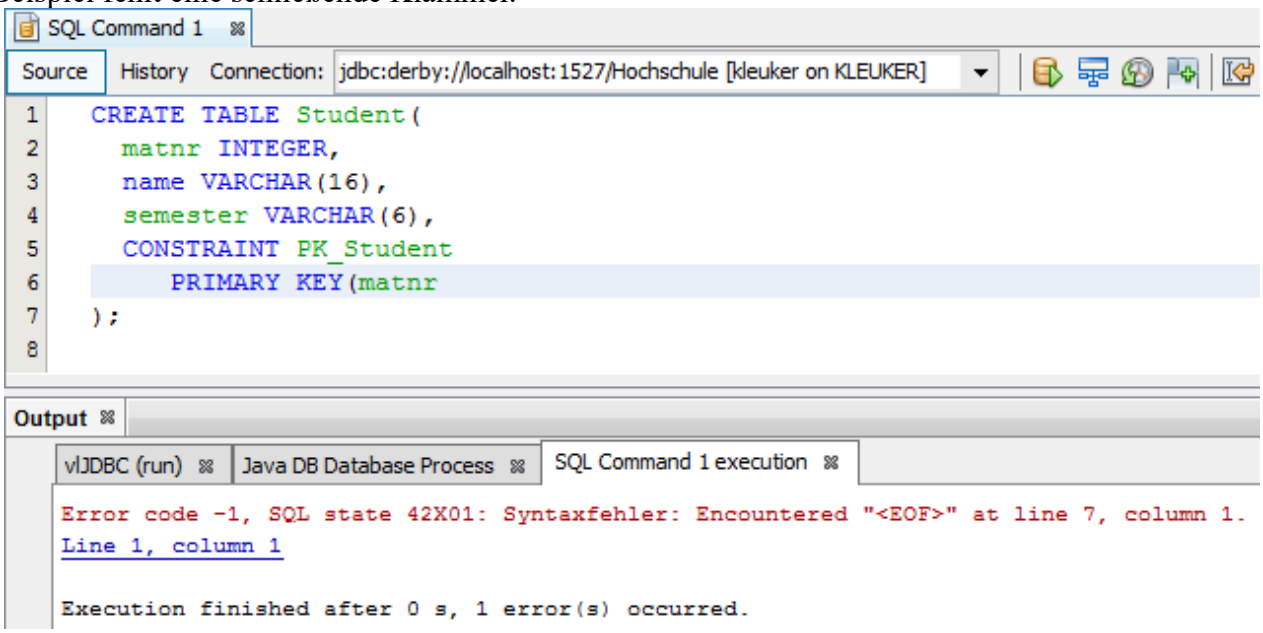
Es öffnet sich dann im Editorenbereich ein Fenster, mit dem beliebige SQL-Befehle, auch ganze Skripte, ausgeführt werden können. Zum Ausführen steht unter anderem das Symbol neben der „Connection“ zur Verfügung, mit dem alle Anweisungen des Eingabefensters ausgeführt werden.



Soll nur eine einzelne Zeile ausgeführt werden, wird das zweite Symbol genutzt. Es wird dann die Zeile ausgeführt, in der der Cursor steht. Der Cursor muss dabei nicht am Anfang der Zeile stehen.



Fehler werden im Output-Bereich unter Nennung des Fensternamens ausgegeben. Im folgenden Beispiel fehlt eine schließende Klammer.

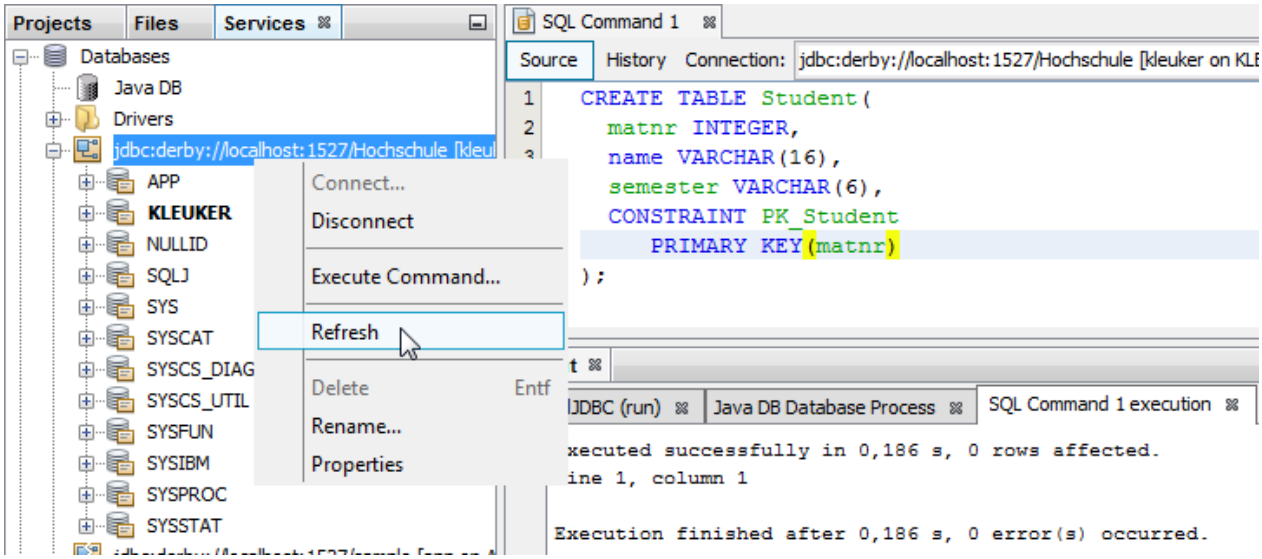


Das Skript wird wie folgt korrigiert.

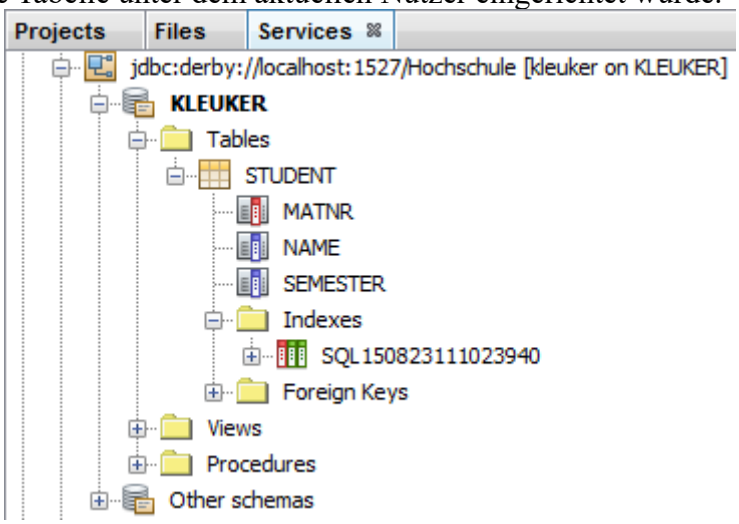
```
CREATE TABLE Student(
  matnr INTEGER,
  name VARCHAR(16),
  semester VARCHAR(6),

  CONSTRAINT PK_Student
    PRIMARY KEY(matnr)
);
```

Nachdem die Tabellenerzeugung erfolgreich abgeschlossen wurde, wird eine Information im Output-Fenster ergänzt. Damit die Tabelle auch im Browser auf der linken Seite angezeigt wird, muss mit einem Rechtsklick auf der Datenbank ein „Refresh“ aufgerufen werden.

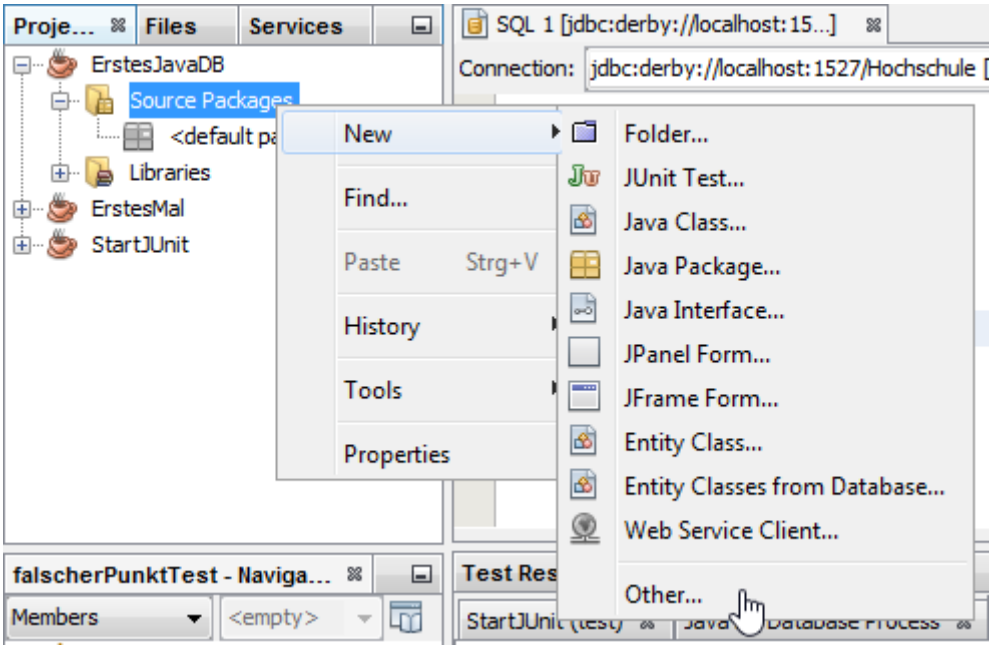


Man erkennt, dass die Tabelle unter dem aktuellen Nutzer eingerichtet wurde.

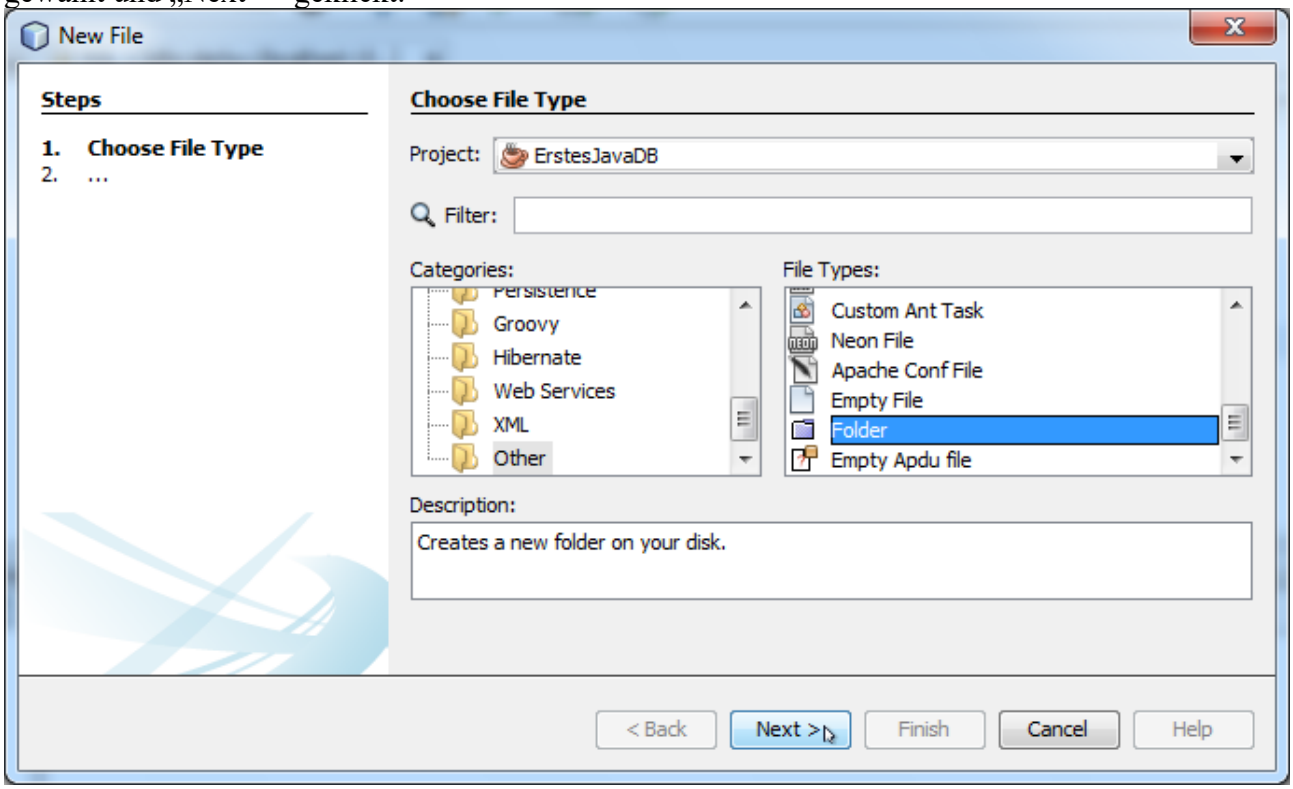


6.4 Speichern und Laden von SQL-Skripten

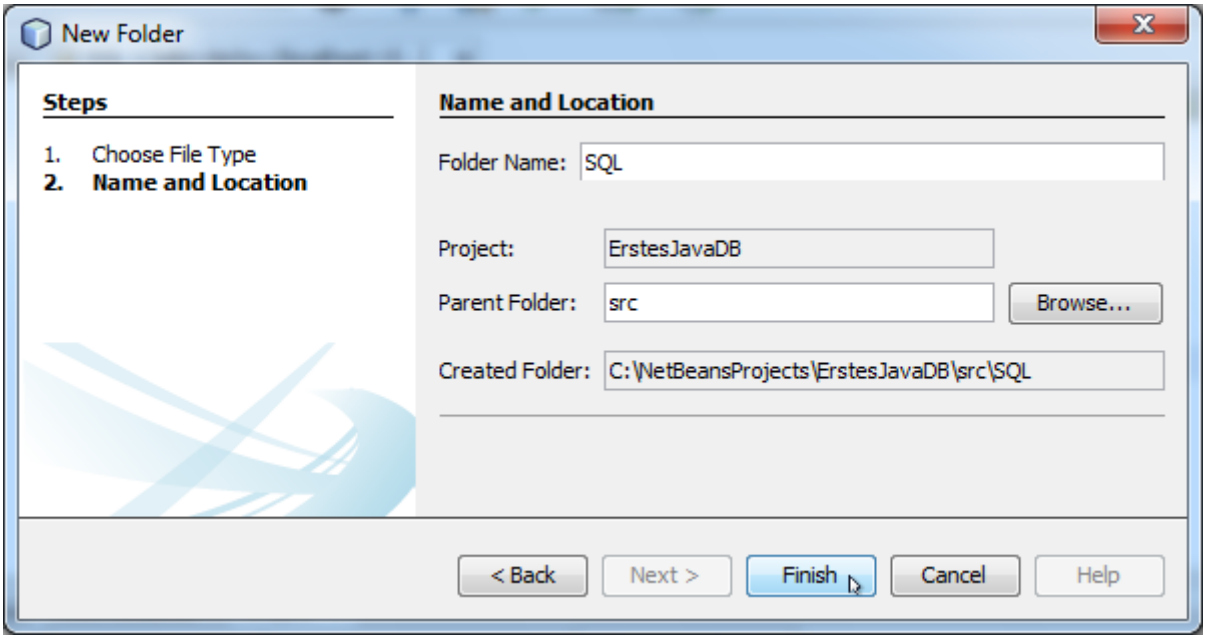
Zum Speichern von SQL-Skripten gibt es verschiedene Ansätze. Ein Ansatz ist, dass dies außerhalb von NetBeans passiert, da die Skripten nicht unmittelbar zur Ausführung eines Programms gehören und so aus einem anderen Editor nach NetBeans kopiert werden. Neue Skripte siund dann wieder manuell in einen anderen Editor zu kopieren, um sie dort abzuspeichern.
Alternativ erfolgt die Speicherung in einem NetBeans-Projekt, was hier exemplarisch gezeigt wird. Dazu wird ein Java-Projekt benutzt, dass entweder im Zusammenhang mit dem SQL-Skript steht oder neu angelegt werden muss.
In dem Projekt wird im Ordner „Source Packages“ ein Unterordner SQL angelegt. Dazu wird ein Rechtsklick auf „Source Package“ gemacht, „New“ und dann „Other…“ geklickt.



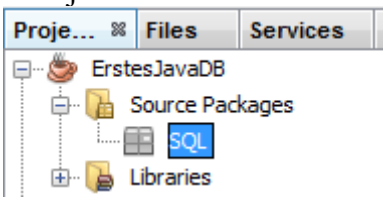
Unter „Categories“ wird unten „Other“ und rechts bei „File Types“ relativ weit unten „Folder“ gewählt und „Next >“ geklickt.



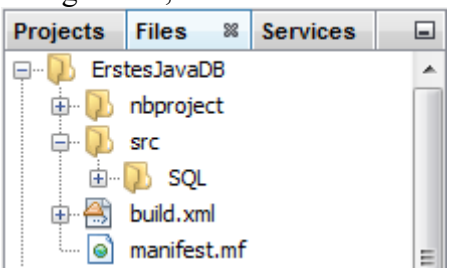
Es wird der „Folder Name“, hier „SQL“ eingegeben und dier Erstellung mit „Finish“ abgeschlossen.



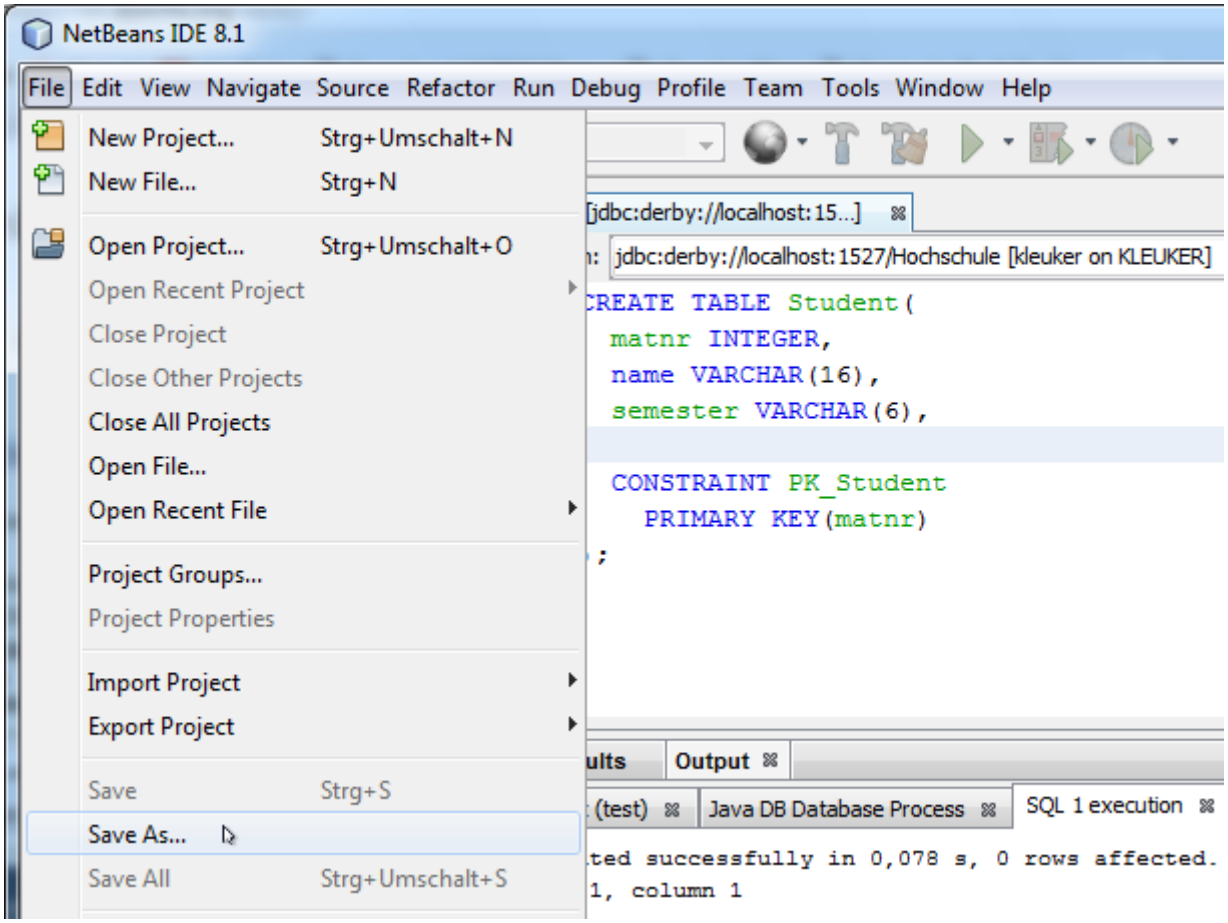
Im Projekt-Browser ist der neue Ordner jetzt sichtbar.



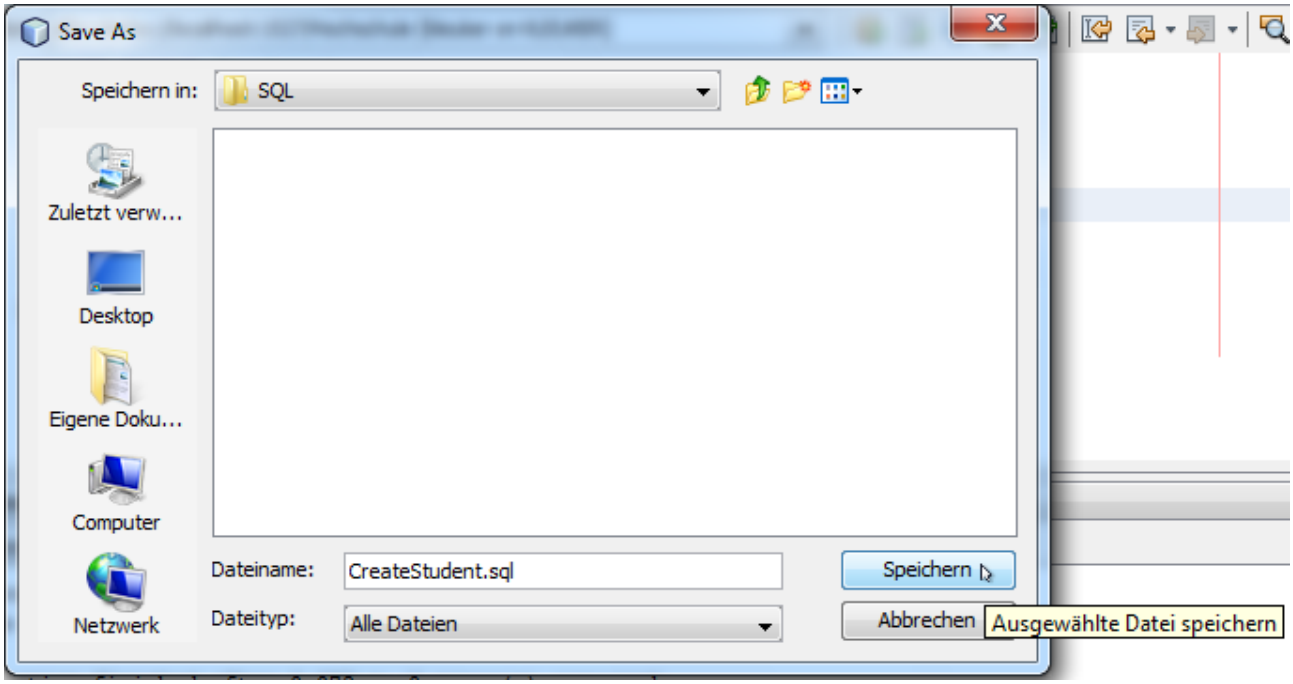
Wird das Projekt im Reiter „Files“ angesehen, wird der neue Ordner im Ordner „src“ angezeigt.



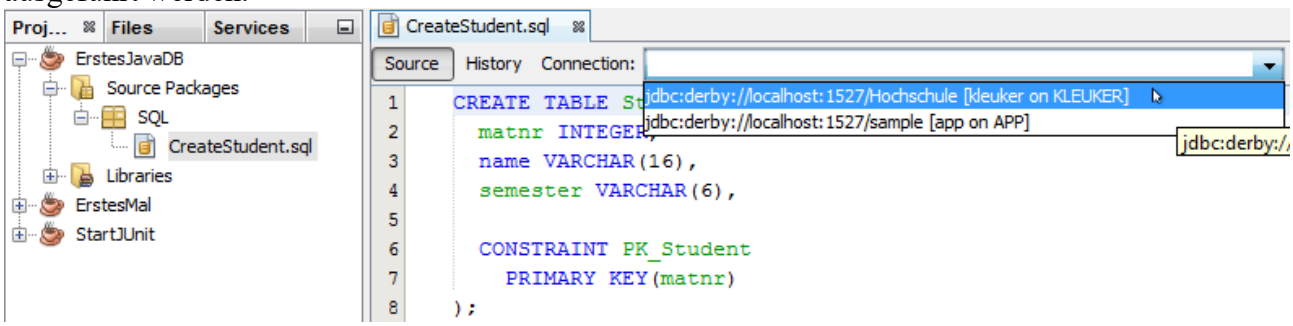
Zum Speichern des SQL-Skripts muss der Cursor irgendwo in dem zu speichernden Skript stehen. Es wird dann „File“ und „Save As…“ ausgewählt.



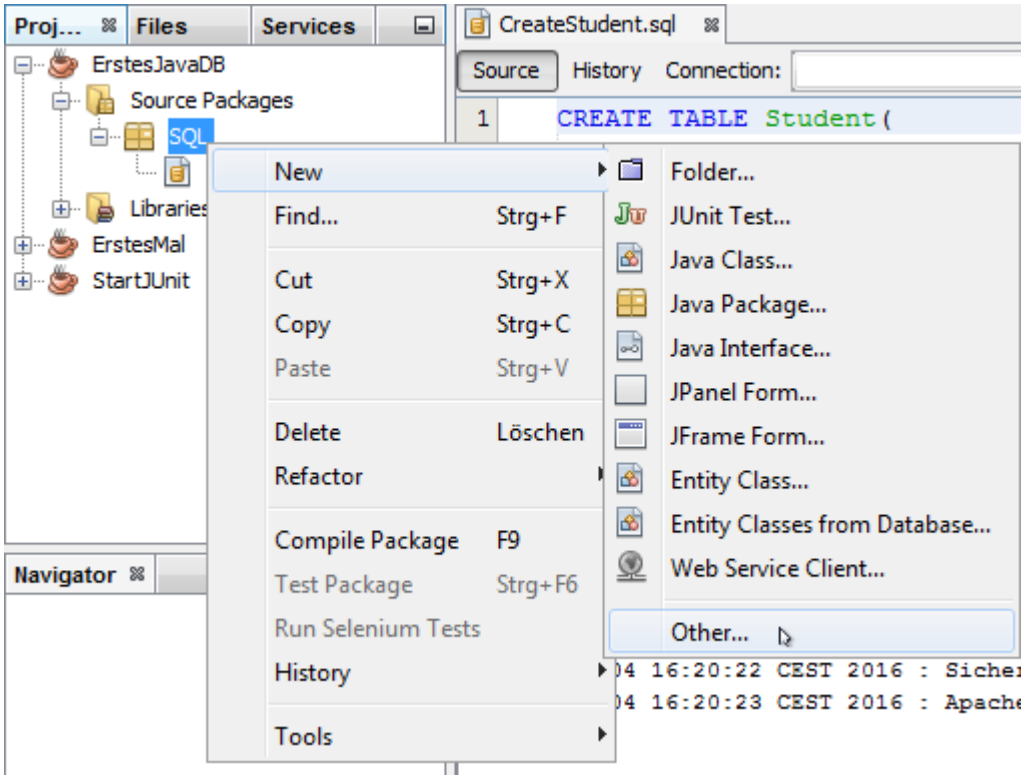
Es wird zu dem neu angelegten Ordner im Projekt gesteuert, der ein Unterordner des Ordners „src“ ist. Es wird der Dateiname eingegeben, der mit „.sql“ endet und „Speichern“ geklickt.



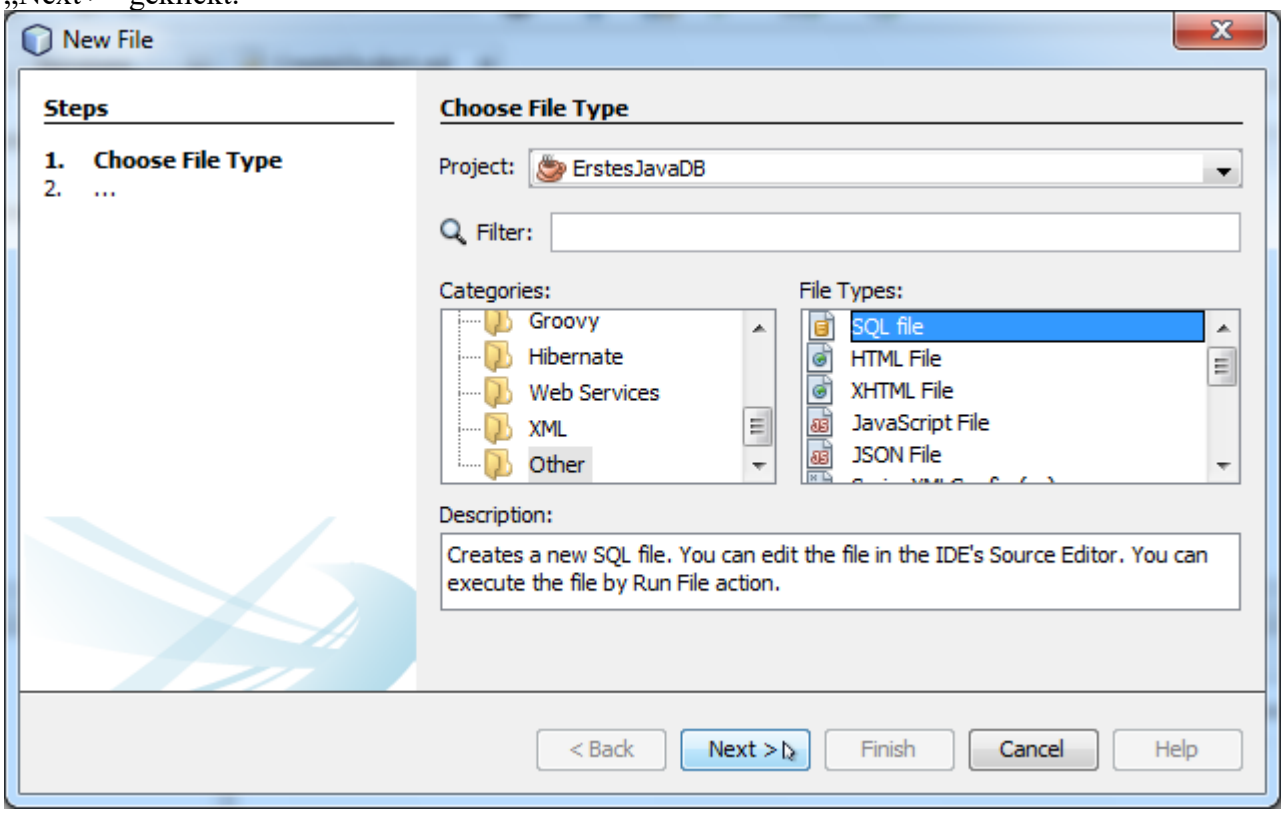
Im Projektordner ist die Datei sichtbar und kann später durch ein Doppelklick geöffnet werden. Hier wird im ersten Schritt in der Drop-Down-Box „Connection“ dann die gewünschte Datenbankverbindung ausgewählt. Danach können die Befehle auf der jetzt verbundenen Datenbank ausgeführt werden.



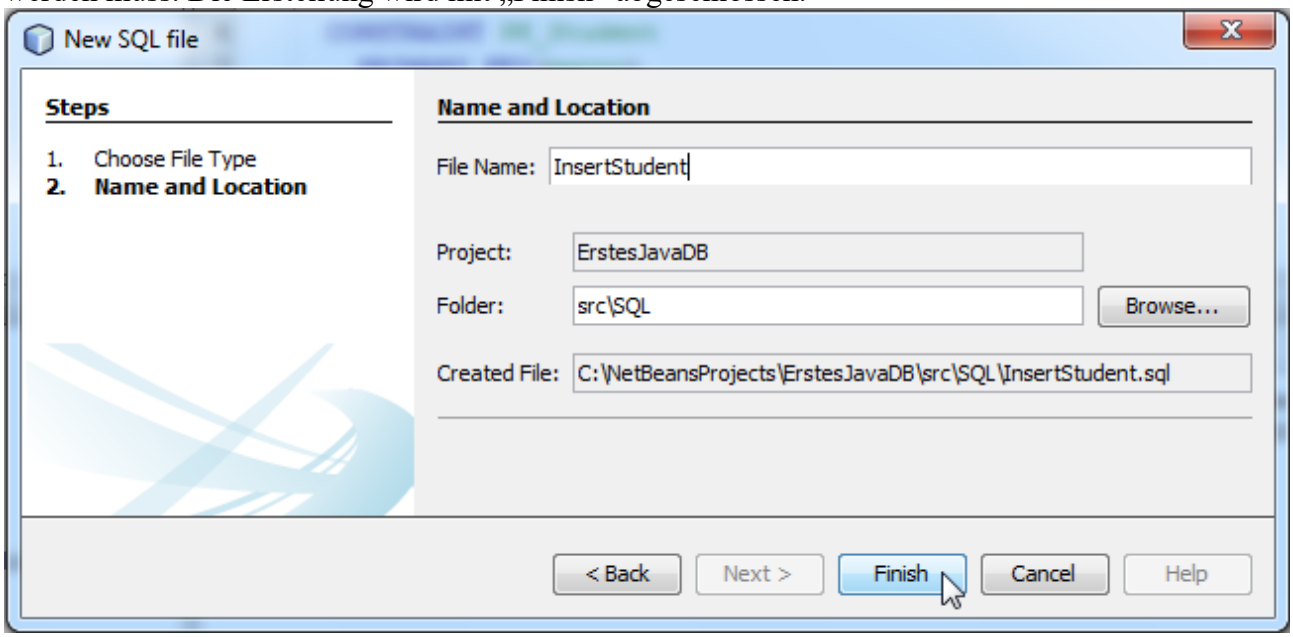
Zum Anlegen eines neuen SQL-Skripts kann als eine Alternative im Projekt ein Rechtsklick auf dem Order „SQL“ gemacht und dann „New“ und „Other…“ gewählt werden.



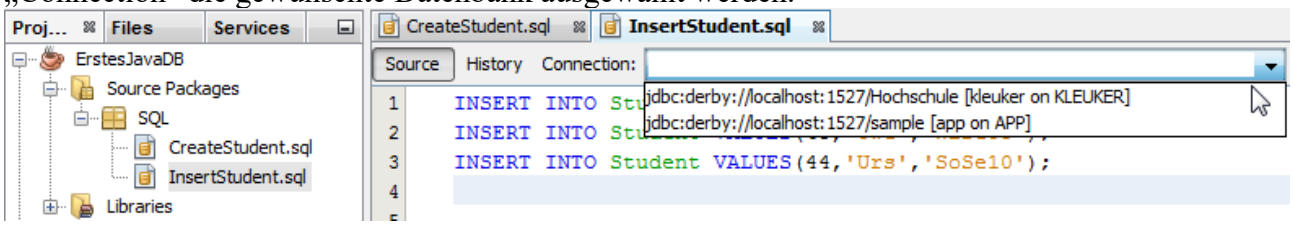
Unter „Categories“ wird unten „Other“ und rechts bei „File Types“ oben „SQL file“ gewählt und „Next >“ geklickt.



Es wird unter „File Name“ der neue Dateiname angegeben, wobei keine Endung „.sql“ angegeben werden muss. Die Erstellung wird mit „Finish“ abgeschlossen.



Bevor die eingegebenen Befehle ausgeführt werden können muss in der Drop-Down-Box „Connection“ die gewünschte Datenbank ausgewählt werden.



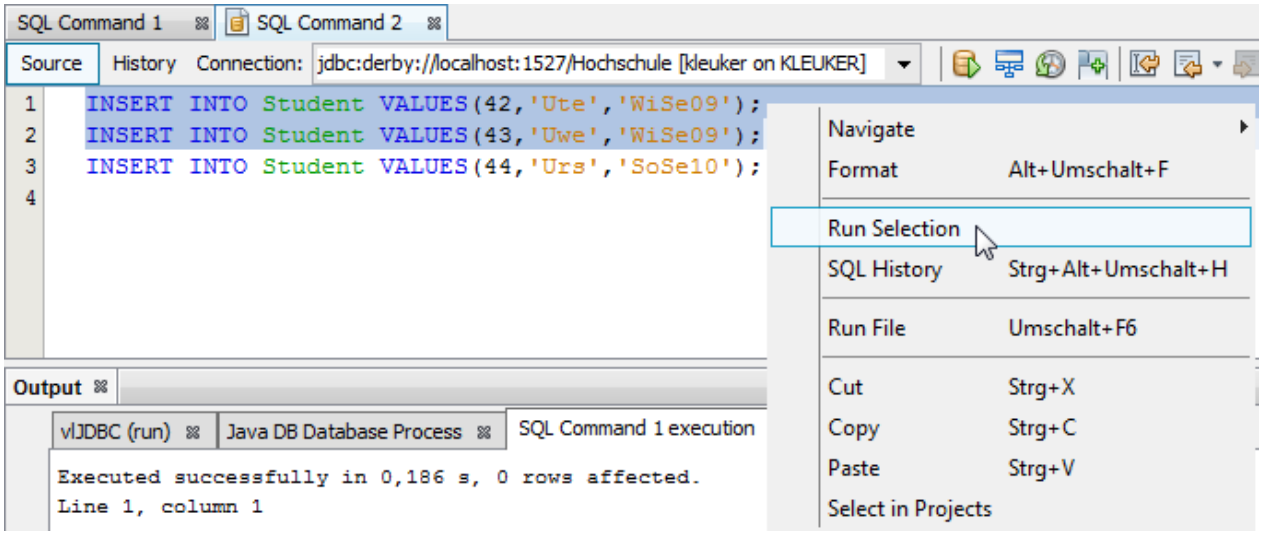
6.5 Ausführung von SQL-Befehlen

Möchte man Datensätze ergänzen, kann man dies entweder durch einen Rechtsklick auf der Tabelle einleiten, oder direkt SQL-Befehle im Editor-Fenster eingeben.

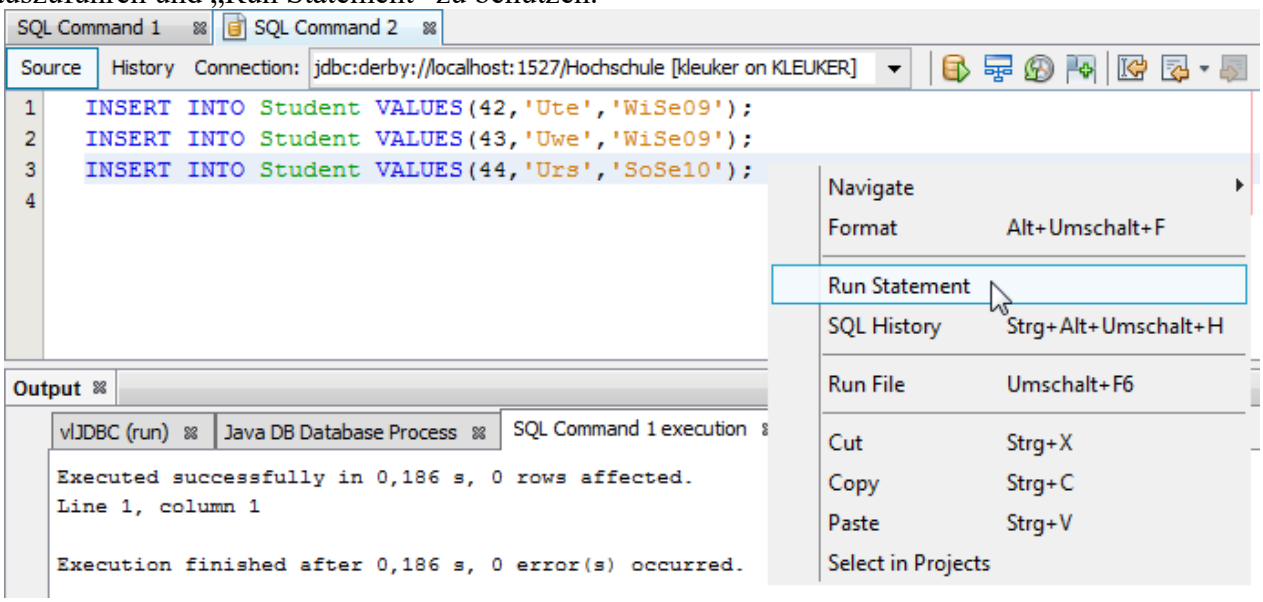
```
INSERT INTO Student VALUES(42,'Ute','WiSe09');
INSERT INTO Student VALUES(43,'Uwe','WiSe09');
INSERT INTO Student VALUES(44,'Urs','SoSe10');
```

Man kann dabei durchaus mit mehreren Fenstern arbeiten, um die Übersicht zu behalten. Es wird also wieder „Execute Command“ ausgewählt.

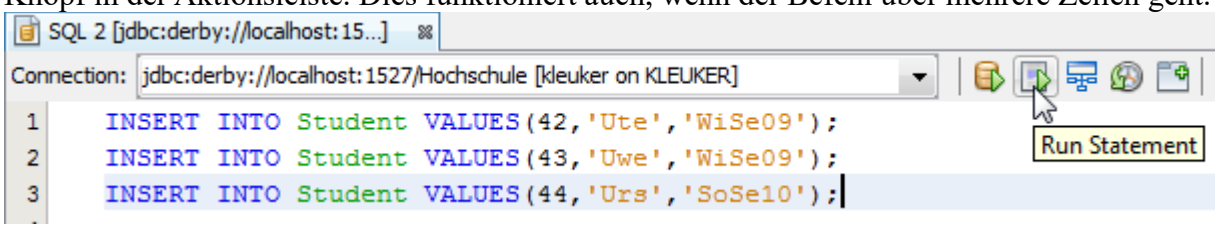
Im Editor besteht dazu auch unter anderem die Möglichkeit Zeilen zu markieren und mit einem Rechtsklicküber „Run Selection“ auszuführen. Die Zeilen müssen dabei genau markiert werden, da genau der Inhalt der Markierung ausgeführt wird.



Alternativ besteht auch die Möglichkeit, irgendwo in einem SQL-Befehl einen Rechtsklick auszuführen und „Run Statement“ zu benutzen.

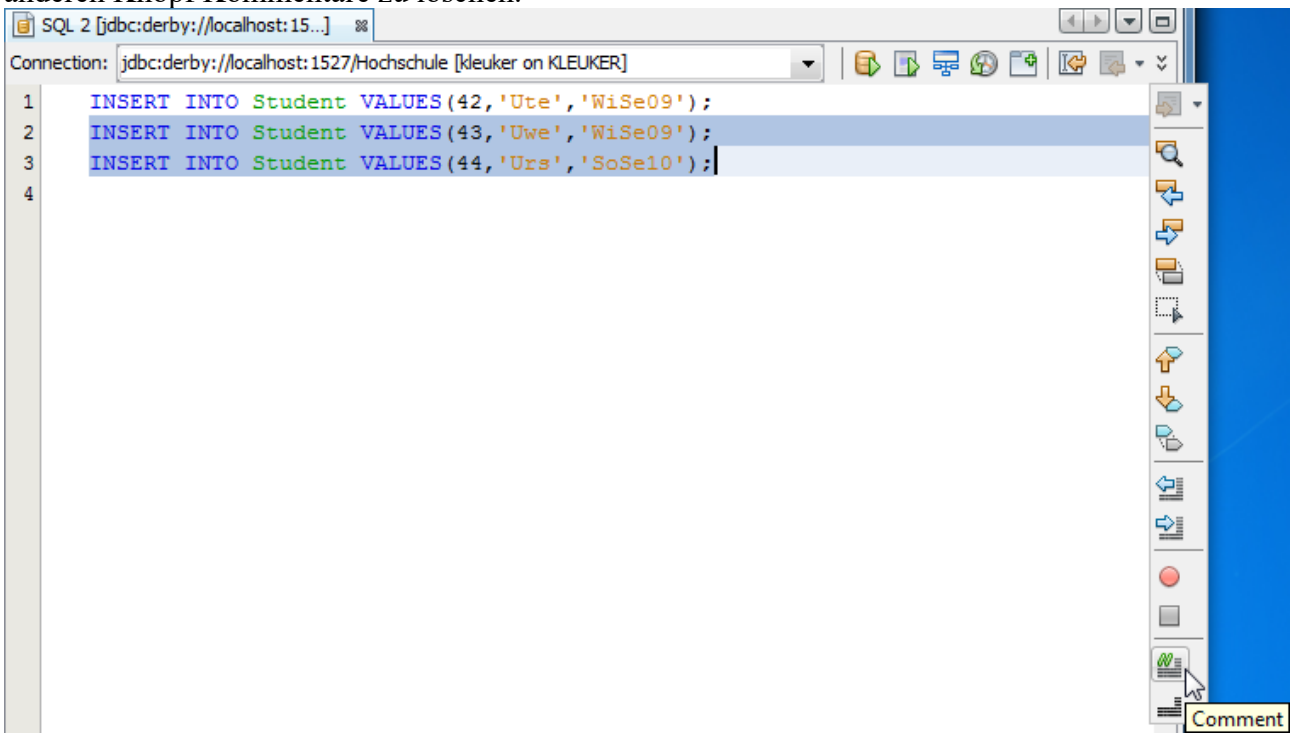


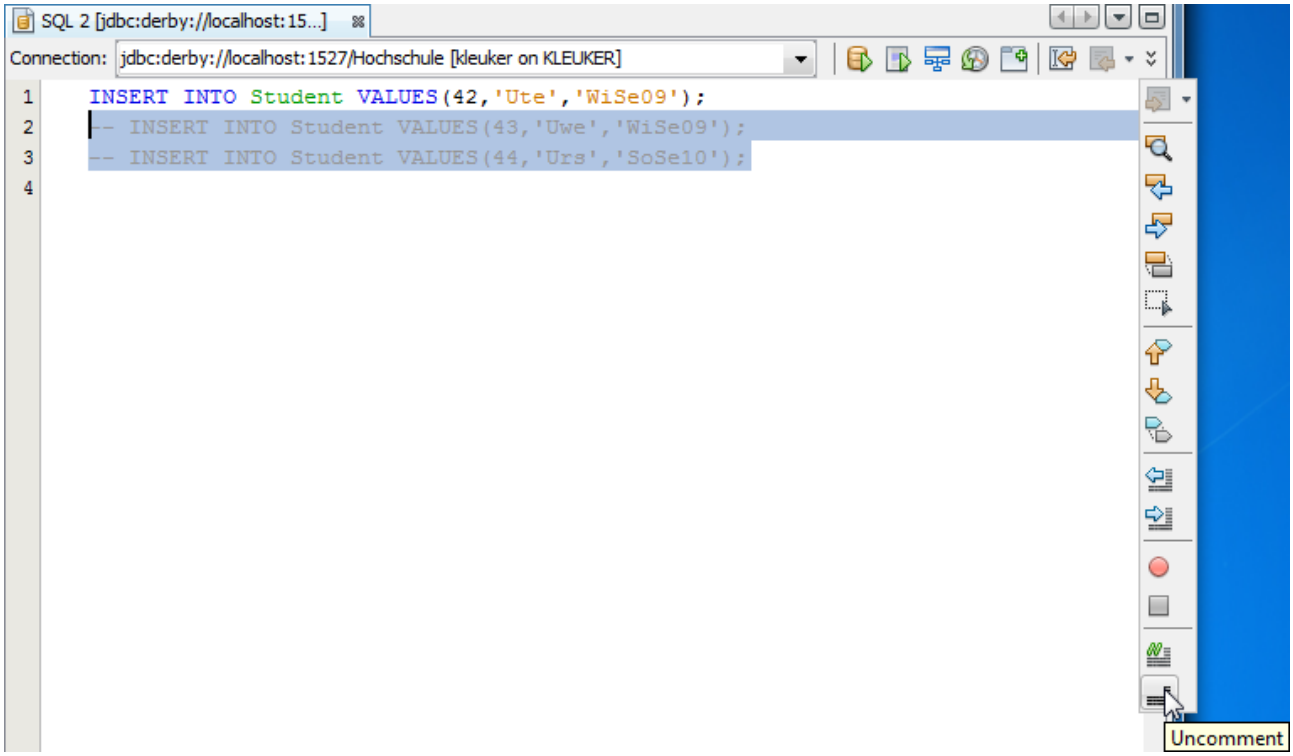
Wenn man den Befehl ausführen möchte, in dem der Cursor steht, gibt es weiterhin den zweiten Knopf in der Aktionsleiste. Dies funktioniert auch, wenn der Befehl über mehrere Zeilen geht.



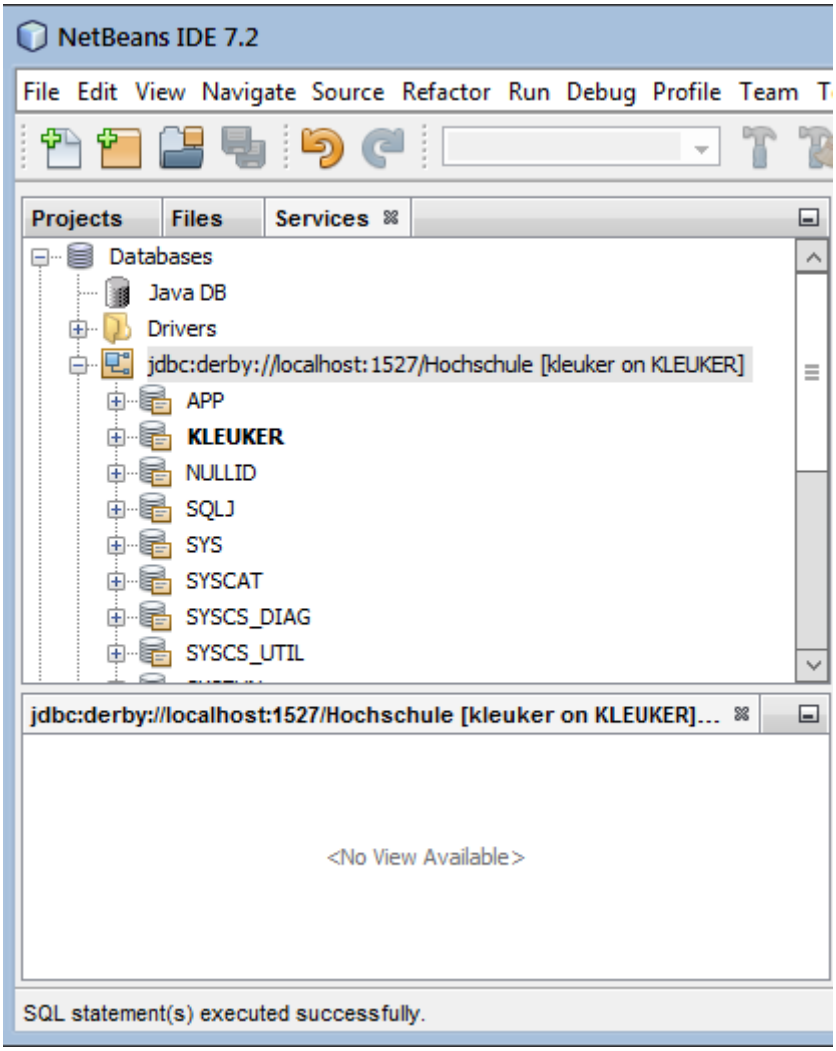
In der Aktionsleiste gibt es weiterhin ganz rechts, bzw. rechts, wenn mit der Maus über den nach unten zeigenden Pfeil gesteuert wird, die Möglichkeit Zeilen auszukommentieren, bzw. mit dem

anderen Knopf Kommentare zu löschen.



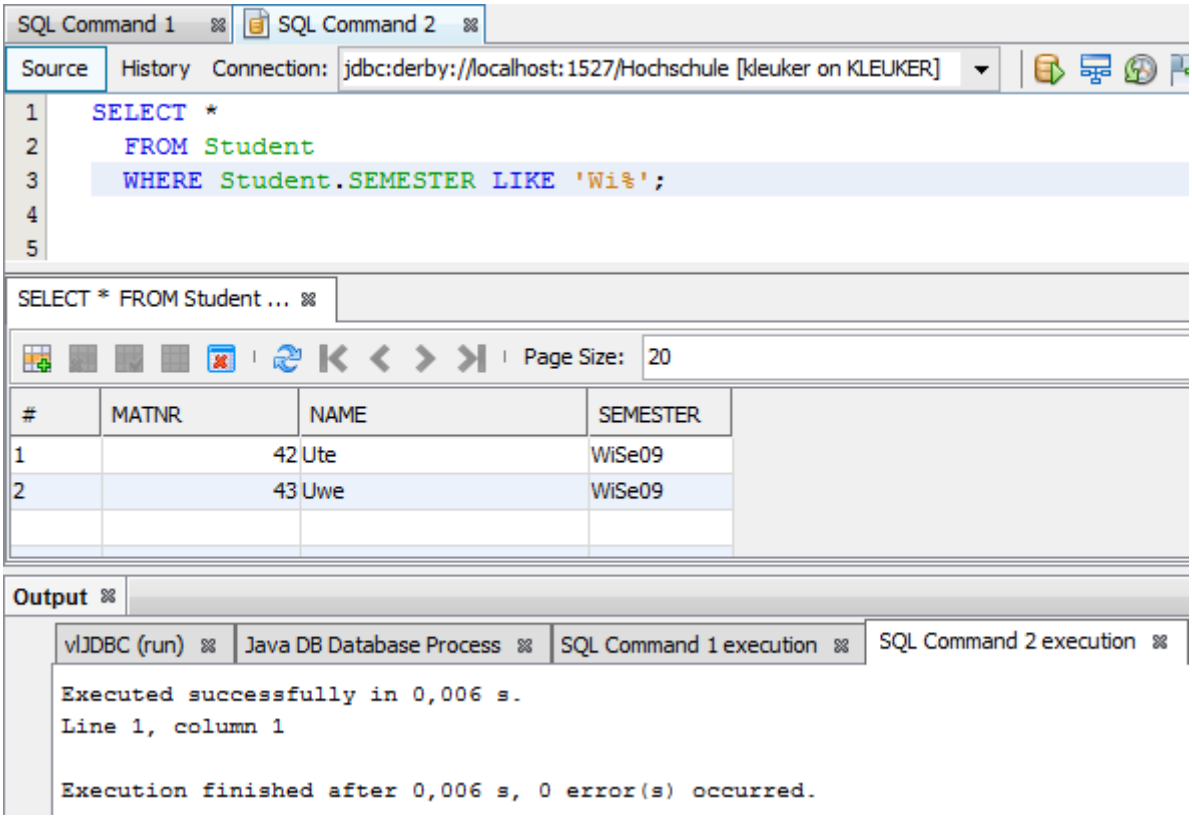


Generell erhält man bei erfolgreich ausgeführten SQL-Befehlen sehr wenig Feedback. Die einzige Information befindet sich häufiger nur ganz unten in der Fußzeile von NetBeans.



Führt man einen SELECT-Befehl aus, wird zwischen dem Editor-Fenster und dem Output-Fenster ein zusätzliches Fenster mit Reitern eingeblendet, die die Ergebnisse enthalten. Evtl. muss man die Kanten zwischen den Fenstern etwas verschieben, damit die Ergebnisse sichtbar werden.

```
SELECT *
  FROM Student
  WHERE Student.SEMESTER LIKE 'Wi%';
```

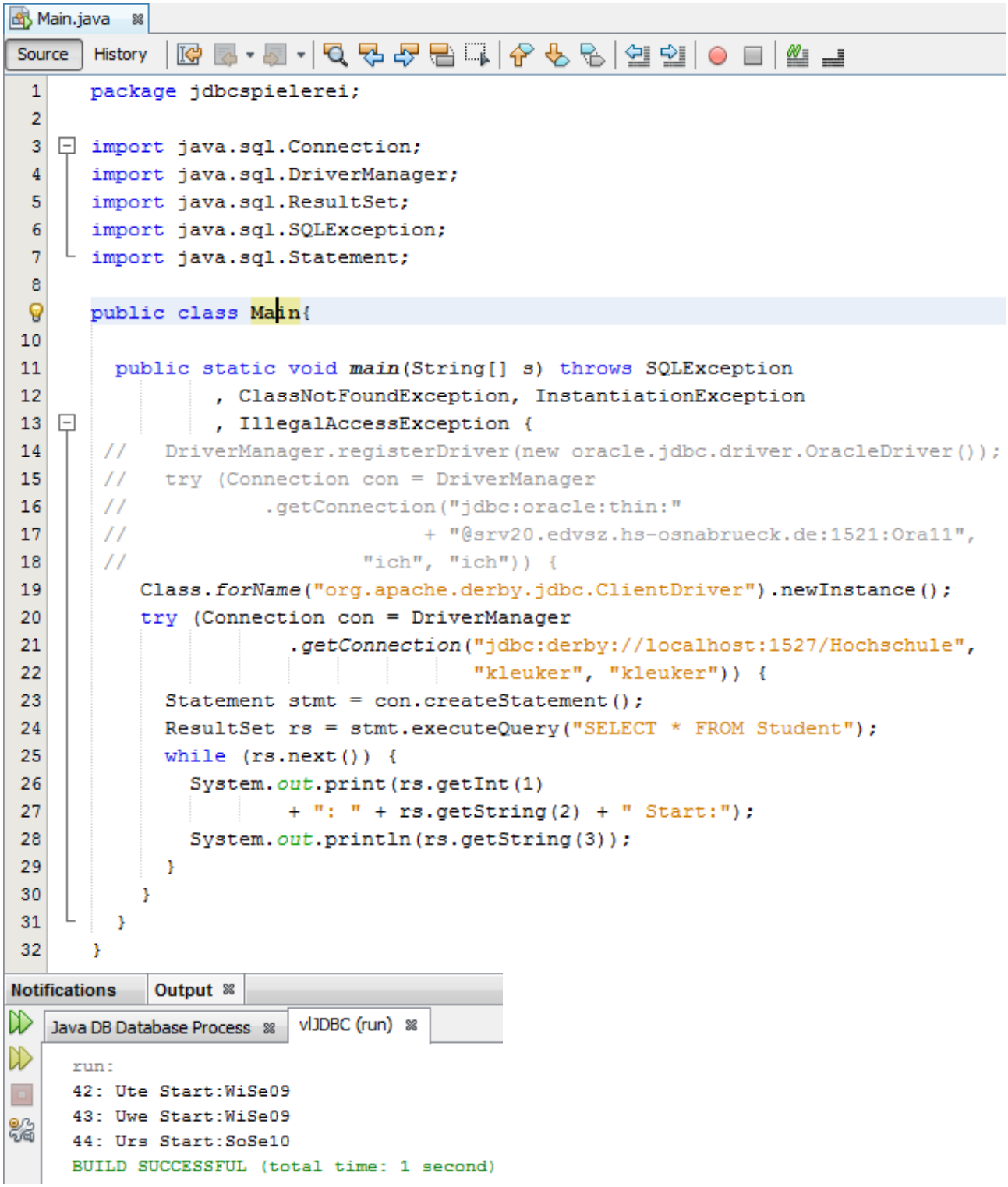


6.6 Zugriff mit JDBC

Der Zugriff mit Hilfe von JDBC erfolgt mit den zugehörigen Sprachkonstrukten, der „Connection-String“ wurde bereits bei jedem SQL-Command-Fenster ausgeben. Ein Beispielprogramm sieht mit zugehöriger Ausgabe wie folgt aus. Man beachte, dass in den Kommentaren die Verbindung zu einer externen Oracle-Datenbank beschrieben ist, für die man zunächst einen JDBC-Treiber herunterladen muss. Weiterhin fällt auf, dass für Oracle der Treiber erst registriert werden muss. Dies ist für Java DB nicht der Fall, da der Treiber bereits in NetBeans integriert ist. Damit das Programm auch problemlos auf anderen Rechnern läuft, kann man die Zeile

```
Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
```

ergänzen.




```
package jdbcspielerei;

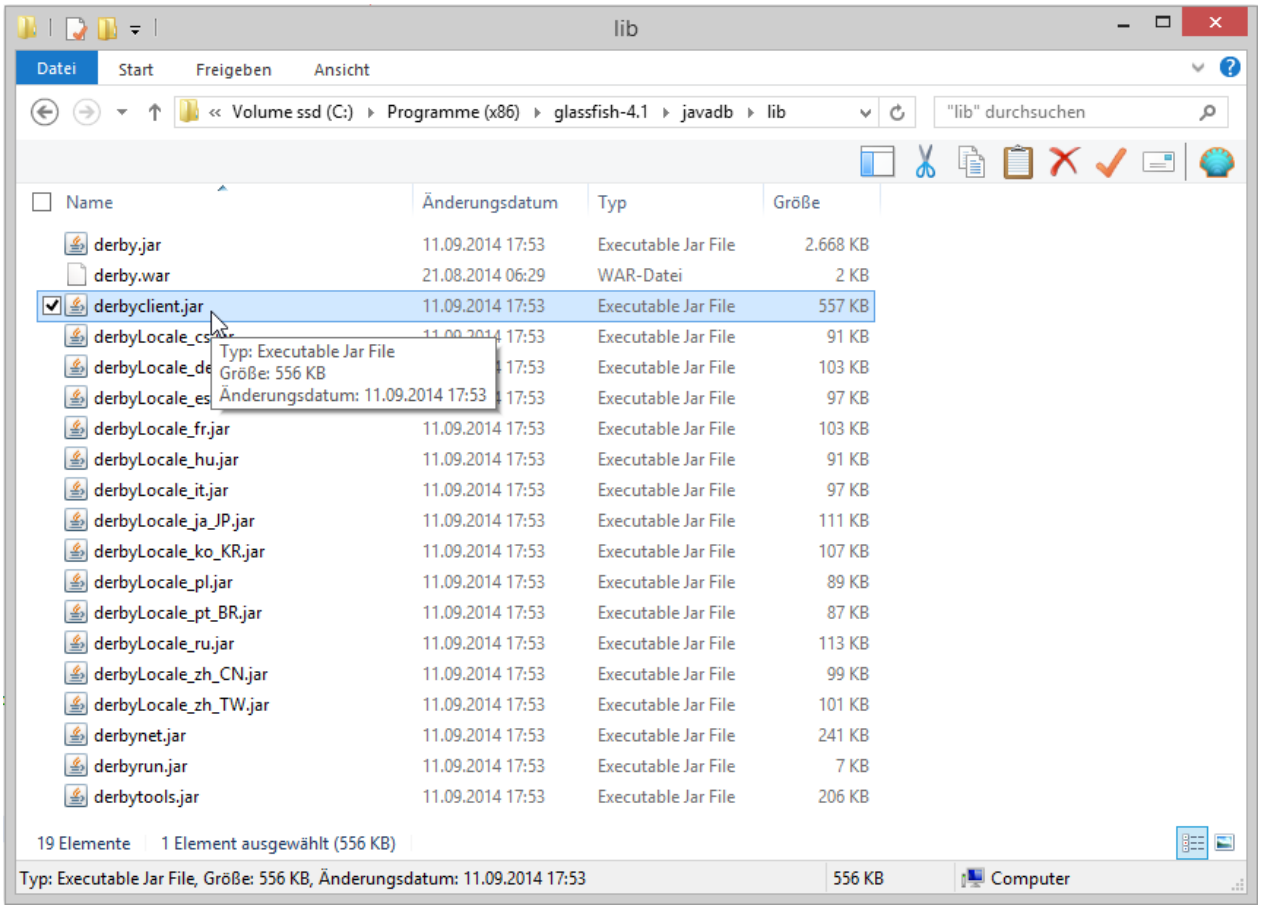
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

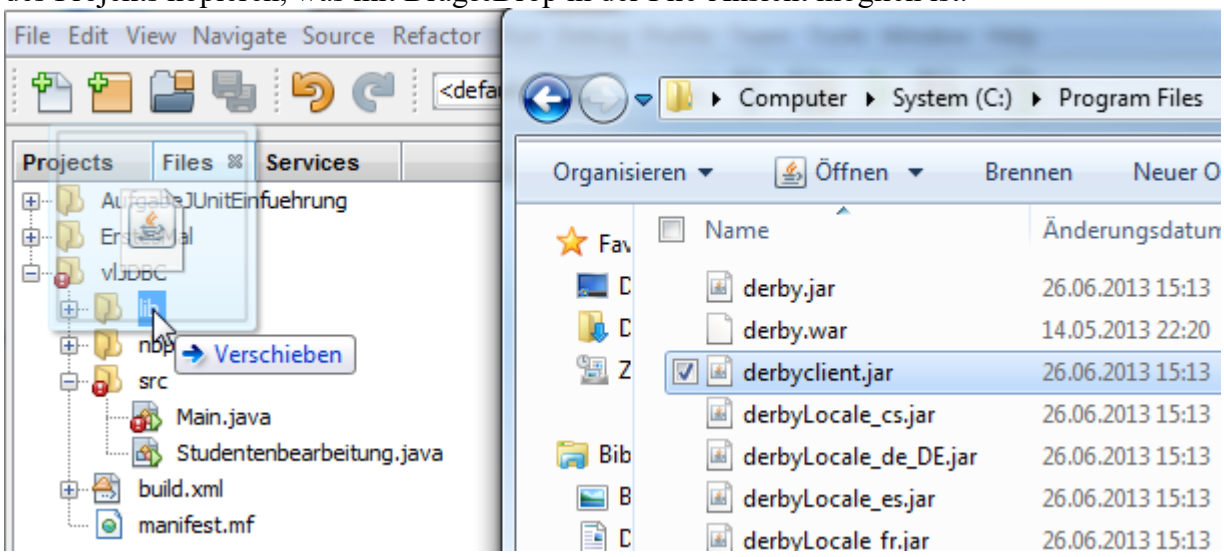
public class Main {

  public static void main(String[] s) throws SQLException
          , ClassNotFoundException, InstantiationException
          , IllegalAccessException {
 //   DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
 //   try (Connection con = DriverManager
 //           .getConnection("jdbc:oracle:thin:"
 //                       + "@srv20.edvsz.hs-osnabrueck.de:1521:Ora11",
 //                   "ich", "ich")) {
    Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
    try (Connection con = DriverManager
                .getConnection("jdbc:derby://localhost:1527/Hochschule",
                               "kleuker", "kleuker")) {
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery("SELECT * FROM Student");
      while (rs.next()) {
        System.out.print(rs.getInt(1)
                + ": " + rs.getString(2) + " Start:");
        System.out.println(rs.getString(3));
      }
    }
  }
}
```

Damit das Projekt erfolgreich ausgeführt werden kann, muss der zur Datenbank passende JDBC-Treiber in das Projekt eingebunden werden. Für Java DB findet man diesen Treiber im Installationsverzeichnis; z.B. C:\Program Files\db-derby-10.14.2.0-bin\lib oder C:\Program Files (x86)\glassfish-4.1\javadb\lib.

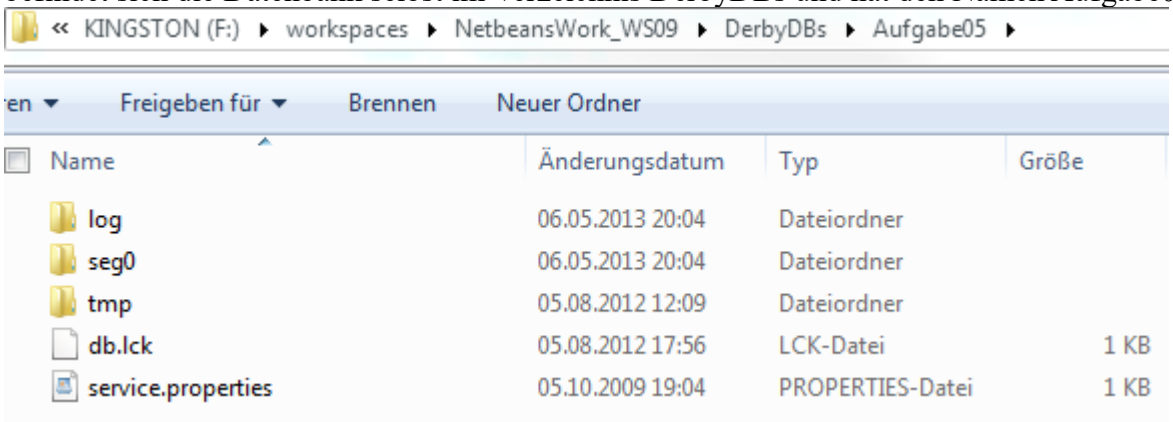


Der benötigte Treiber heißt derbyclient.jar. Die Nutzung von Jar-Dateien wird im Abschnitt „4.6 Nutzung von Bibliotheken/Jar-Files (NetBeans 8)“ beschrieben. Man kann entweder auf die Jar-Datei im angegebenen Ordner referenzieren oder, besser, zunächst den Treiber in ein Unterverzeichnis lib des Projekts kopieren, was mit Drag&Drop in der File-Ansicht möglich ist.



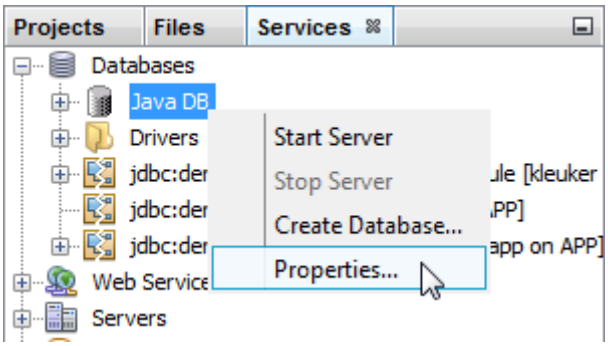
6.7 Einbindung existierender Java DB-Datenbanken

NetBeans erlaubt die flexible Speicherung und erneute Einbindung von Projekten. Ähnlich verhält es sich mit Datenbank-Dateien. Dies ermöglicht z. B. die Speicherung von Datenbanken auf USB-Sticks.
Die relevante Datenbank sei z. B. im Verzeichnis `F:\workspaces\NetBeansWork_WS09\DerbyDBs\Aufgabe05` gespeichert. Dabei befindet sich die Datenbank selbst im Verzeichnis DerbyDBs und hat den Namen Aufgabe05.

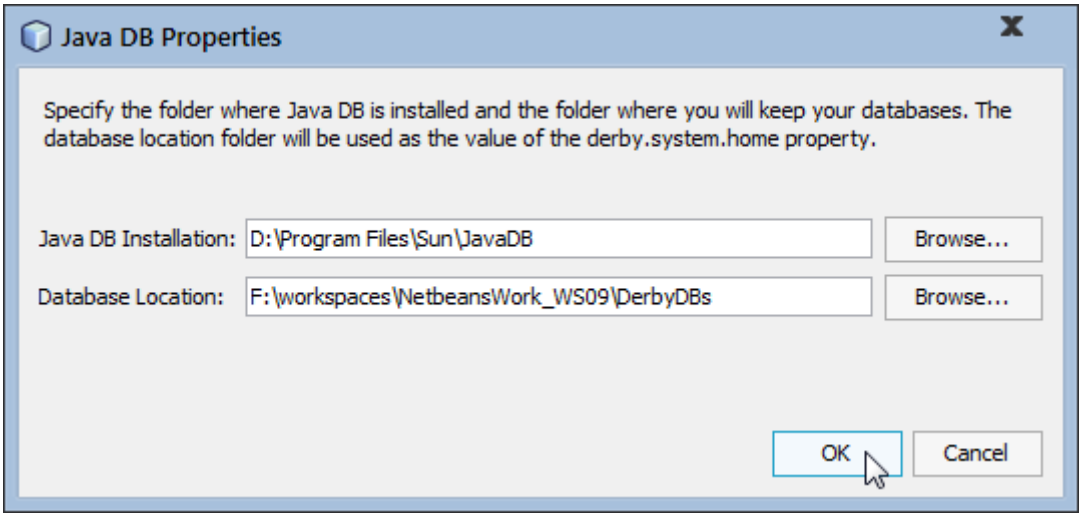


Generell arbeitet Java DB nur direkt mit den Datenbanken, die in der Database Location unter „Properties“, siehe auch die nächsten beiden Abbildungen, eingetragen sind. Gibt es fachlich keine anderen Notwendigkeiten, ist es der einfachste Schritt, die zu nutzende Datenbank in dieses Verzeichnis zu kopieren. Die Datenbank wird dann direkt unter „Java DB“ angezeigt. Dies ist für die meisten Programme kein Problem, da nicht der physikalische Ort der gespeicherten Datenbank, sondern der JDBC-Connection-String relevant ist.

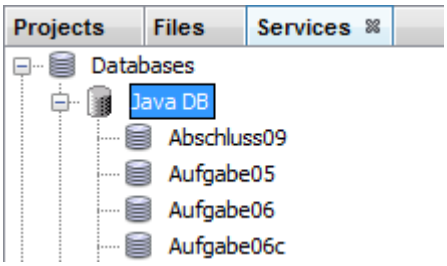
Um die Datenbank direkt in ihrem Ursprungsverzeichnis zu nutzen, muss zunächst der Pfad zur Datenbank im System eingetragen werden. Dazu wird ein Rechtsklick auf „Java DB“ unter dem Reiter Services gemacht.



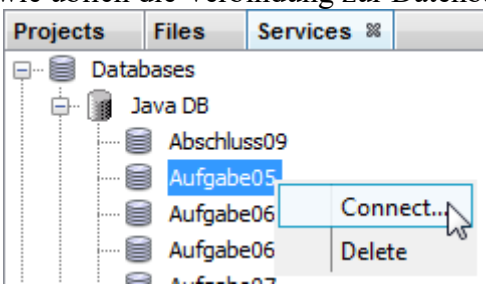
Bei den Properties wird der Pfad zur Datenbank unter „Database Location“ eingetragen.



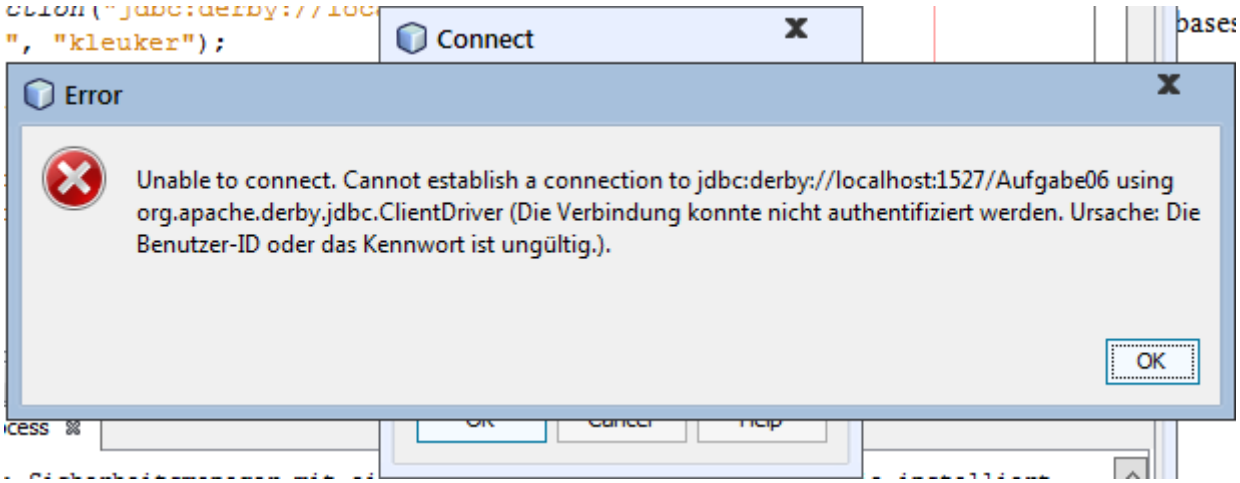
Danach stehen unter Java DB alle im Ordner DerbyDBs vorhandenen Datenbanken zur Verfügung.



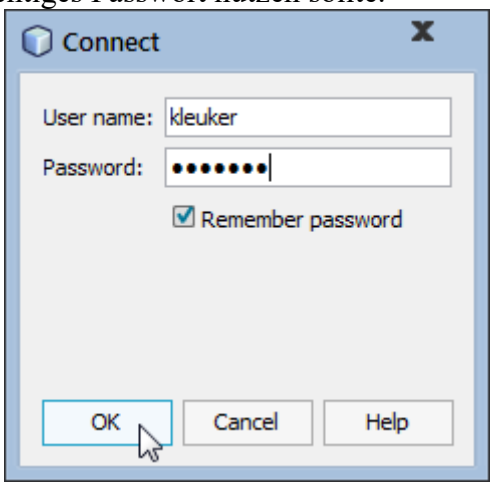
Durch einen Rechtsklick kann wie üblich die Verbindung zur Datenbank aufgebaut werden.



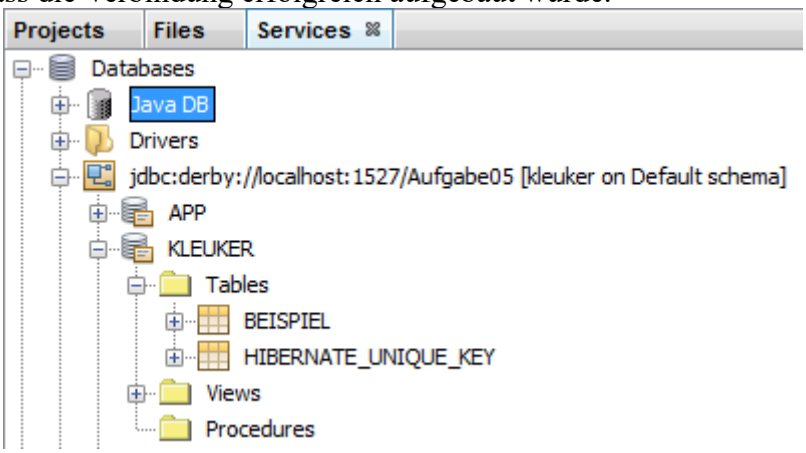
Man sollte jetzt die Nutzer und Passwort-Daten noch wissen. Die folgende Fehlermeldung zeigt nur, dass ohne weitere Zugangsinformationen keine Verbindung aufgebaut werden kann.



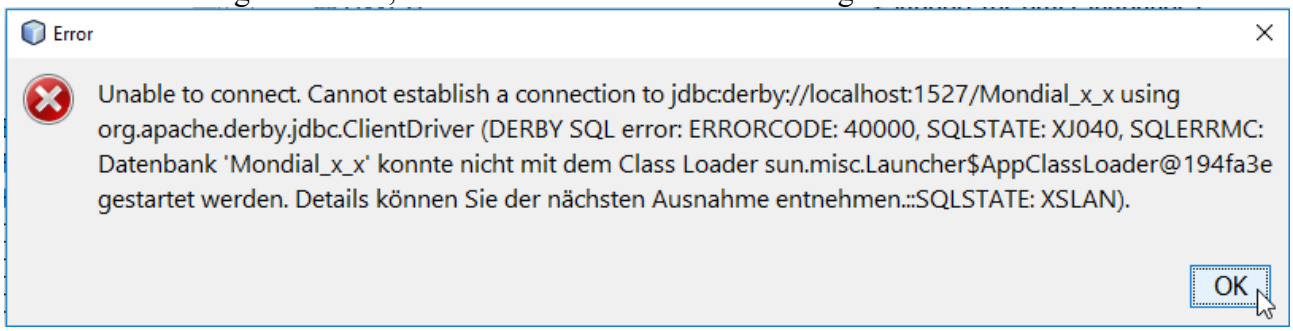
Die Meldung kann man einfach schließen, danach muss der Nutzer und das Passwort eingegeben werden. Sollte weder Nutzer noch Passwort eingetragen sein, klickt man einfach auf „OK“. Lässt man den Haken bei „Remember Password“, wird man nicht wieder gefragt. Wie immer gilt, dass man beim Experimentieren kein wichtiges Passwort nutzen sollte.



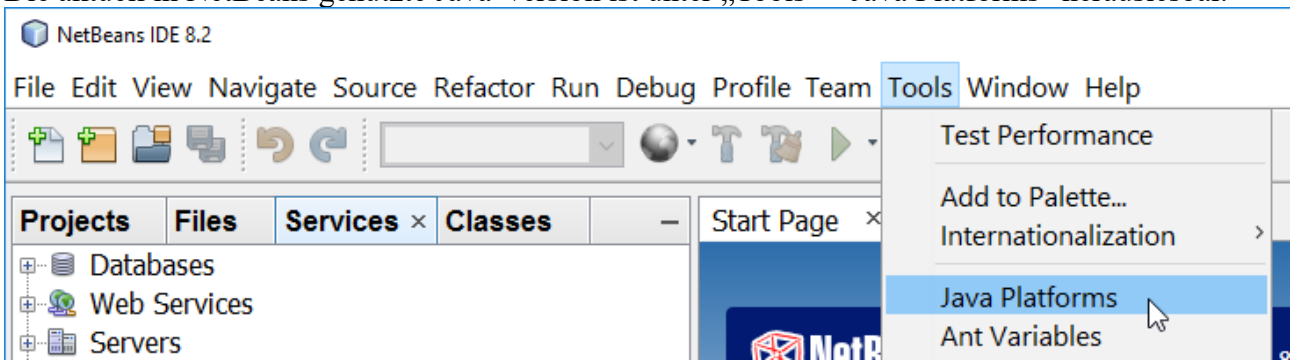
Man sieht dann, dass die Verbindung erfolgreich aufgebaut wurde.



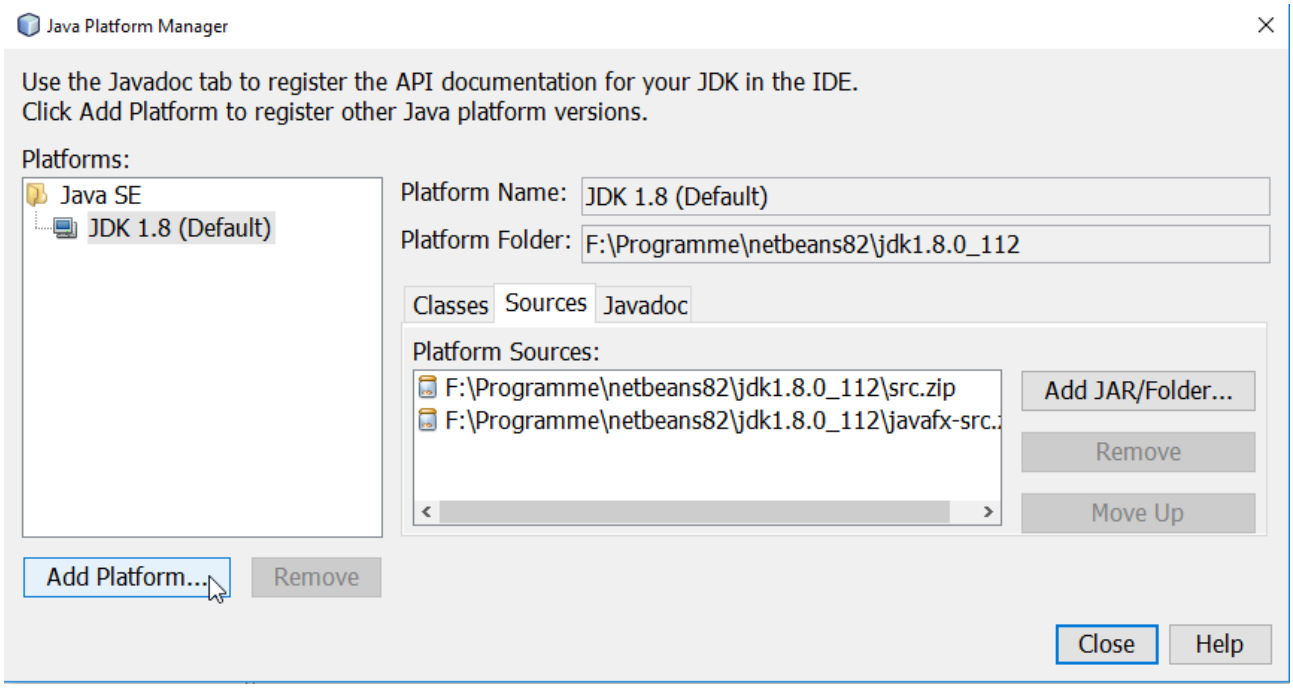
Das Kopieren von Datenbanken kann scheitern, wenn unterschiedliche Datenbank-Versionen genutzt werden. Dies ist dann problematisch, wenn die neu einzubindende Datenbank mit einem neueren System erstellt wurde, als das, was gerade für die Datenbank genutzt wird. Die kritische Meldung hat dann meist die folgende Form, die leifder keinen Schluss auf die genaue Problematik zulässt..



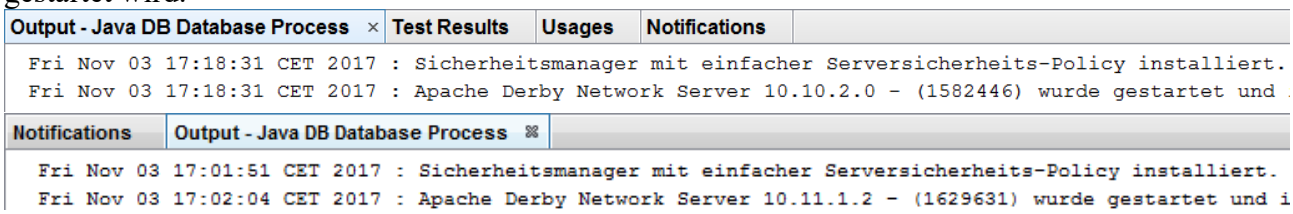
Die aktuell in NetBeans genutzte Java-Version ist unter „Tools -> Java Platforms“ herauslesbar.



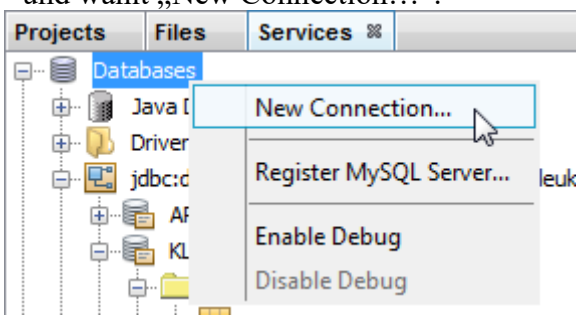
Hier kann auch ein weiteres JDK installiert werden, wie es in „4.7 Installation einer neueren/weiteren Java-Version“ beschrieben ist.



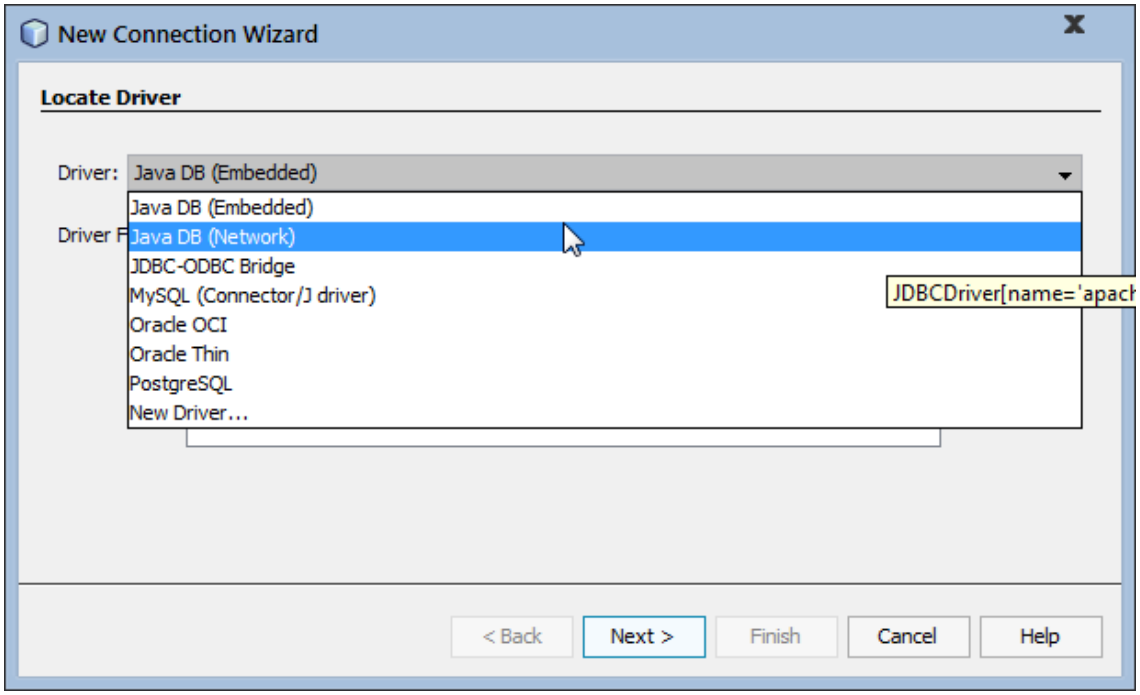
Über die verwendete Datenbank-Version wird unten im Reiter „Output – JavaDB Databes Process“ informiert. Bei Bedarf wird die Meldung sichtbar, wenn die Datenbank gestoppt und dann wieder gestartet wird.



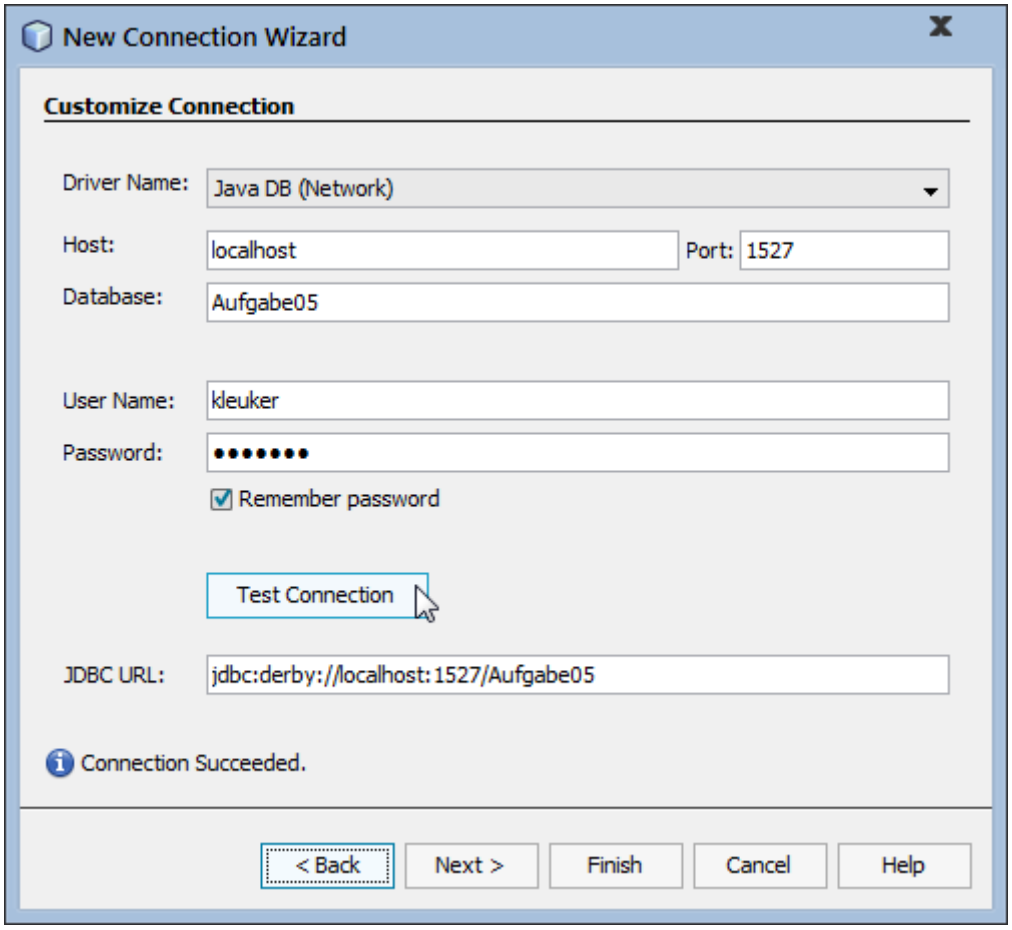
Will man die Verbindung zu anderen existierenden Datenbanken aufbauen, macht man einen Rechtsklick auf „Databases“ und wählt „New Connection…“.



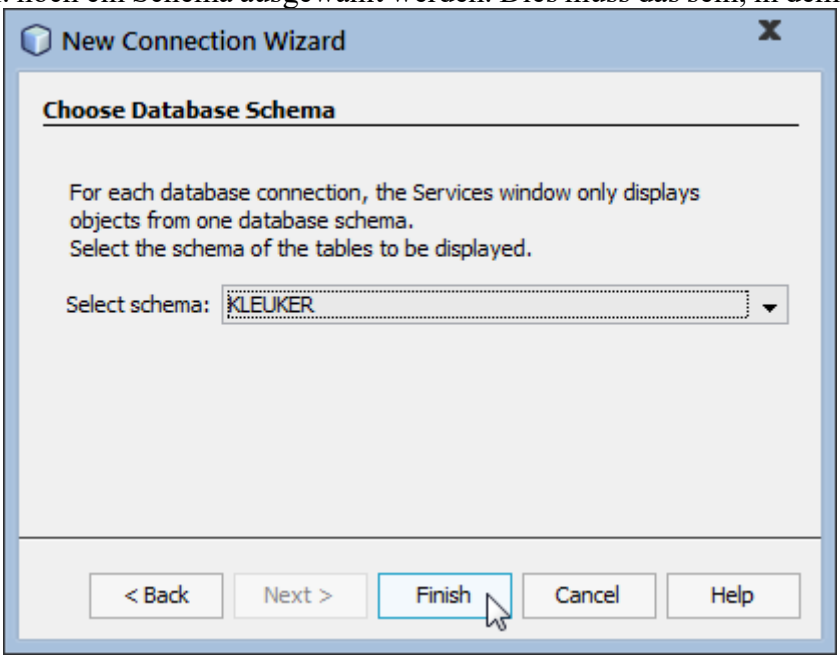
Dann muss oben der passende Treiber ausgewählt werden, NetBeans hat bereits einige installiert, hier können weitere integriert werden. Nach der Auswahl wird „Next>“ gedrückt.



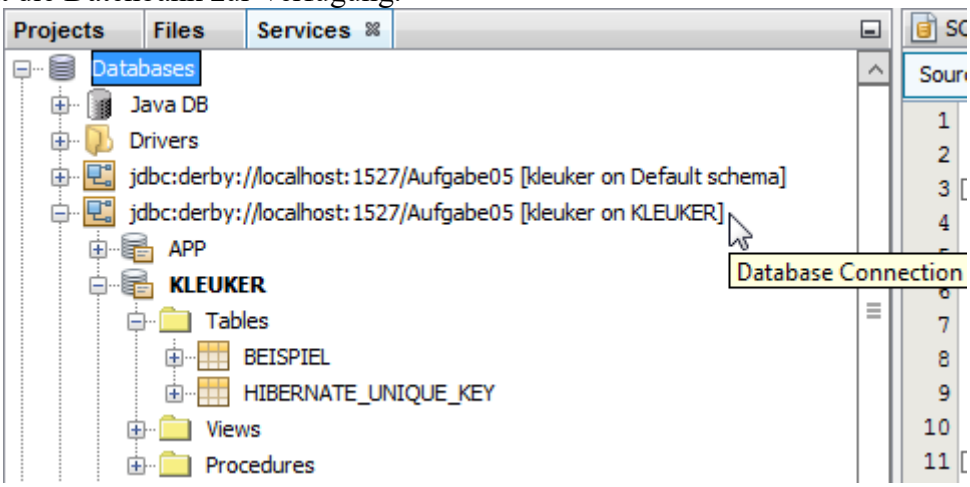
Hier müssen jetzt die Verbindungsdaten eingetragen werden. Vereinfachend wird die gerade vorher eingebundene Datenbank genutzt. Man beachte die Möglichkeit über „Test Connection“ zu prüfen, ob eine erfolgreiche Datenbankverbindung aufgebaut werden kann. Danach wird „Next>“ gedrückt.



Danach muss evtl. noch ein Schema ausgewählt werden. Dies muss das sein, in dem die Daten liegen.



Danach steht die Datenbank zur Verfügung.



6.8 Stored Procedures und Trigger

Mit Hilfe von Stored Procedures kann man Datenbanken „serverseitig“, also innerhalb der Datenbank erweitern. Trigger ermöglichen es, auf INSERT, UPDATE und DELETE-Befehle zu reagieren und diese z. B. nach einer Prüfung der Aktion abzulehnen. In Java DB werden die Erweiterungen in Java basierend auf Klassenmethoden umgesetzt.
Das hier gezeigte Beispiel nutzt die folgende Gebotstabelle.

```
CREATE TABLE Gebot(
  mnr INTEGER, -- Mitgliedsnummer
  ware INTEGER, -- Identifikator der Ware
  gebot DECIMAL(8, 2), -- gebotener Preis
  PRIMARY KEY(mnr,ware,gebot)
);
```

Umzusetzen ist die Anforderung, bei neuen Geboten (insert oder update erlaubt) für die gleiche Ware muss das eigene Gebot (gleiche mnr) erhöht werden.
Zunächst wird eine Java-Methode geschrieben, die folgende Randbedingungen erfüllt:

- es ist eine Klassenmethode (static)
- der Rückgabetyp ist void
- die Parameter entsprechen der zu untersuchenden Aufgabe, hier wird ein Gebot bestehend aus seinen Parametern übergeben
- im Fehlerfall wird eine eigene SQLException geworfen, die als ersten Parameter den Grund und als zweiten Parameter ein individuell aus dem Intervall 30000 bis 38000 gewählten SQL-State hat.

```
package gebottrigger;

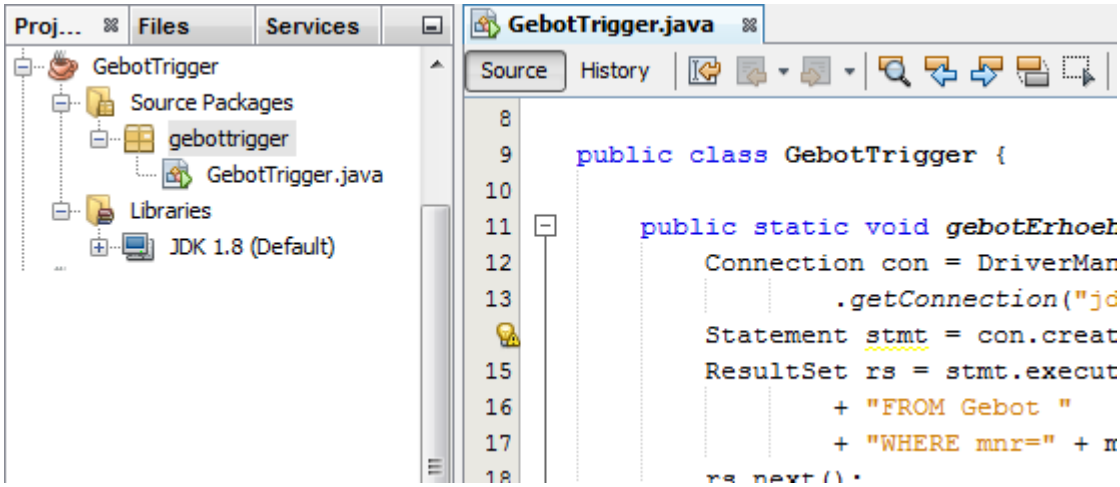
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.sql.Statement;

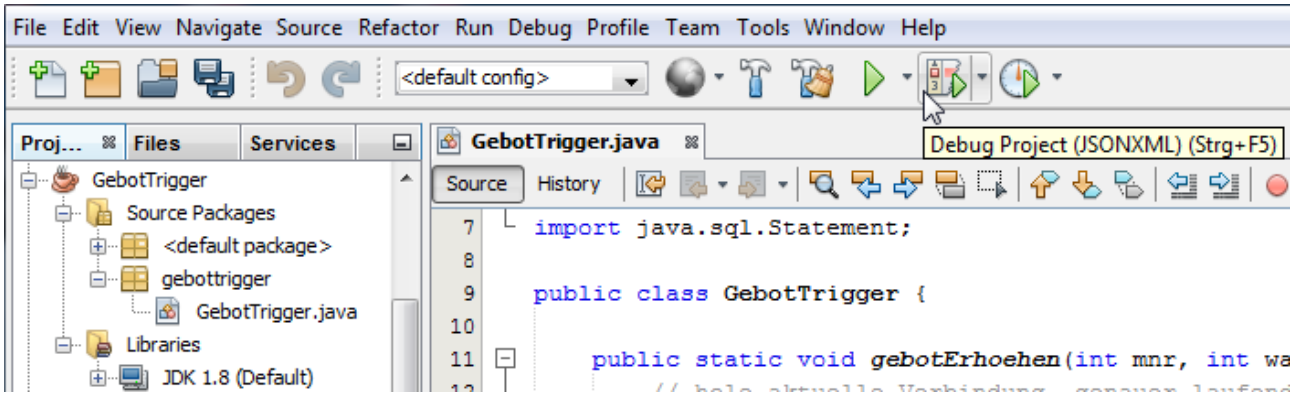
public class GebotTrigger {

   public static void gebotErhoehen(int mnr, int ware, double gebot) throws SQLException {
        // hole aktuelle Verbindung, genauer laufende Transition
        Connection con = DriverManager
                .getConnection("jdbc:default:connection");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT MAX(gebot) "
                + "FROM Gebot "
                + "WHERE mnr=" + mnr + " AND ware=" + ware);
        rs.next();
        double max = rs.getDouble(1);
        System.out.println(rs.wasNull() + " " + max);
        if (!rs.wasNull() && max >= gebot) {
            throw new SQLException("Gebot erhohen!", "30009");
        }
        stmt.close();
    }
}
```

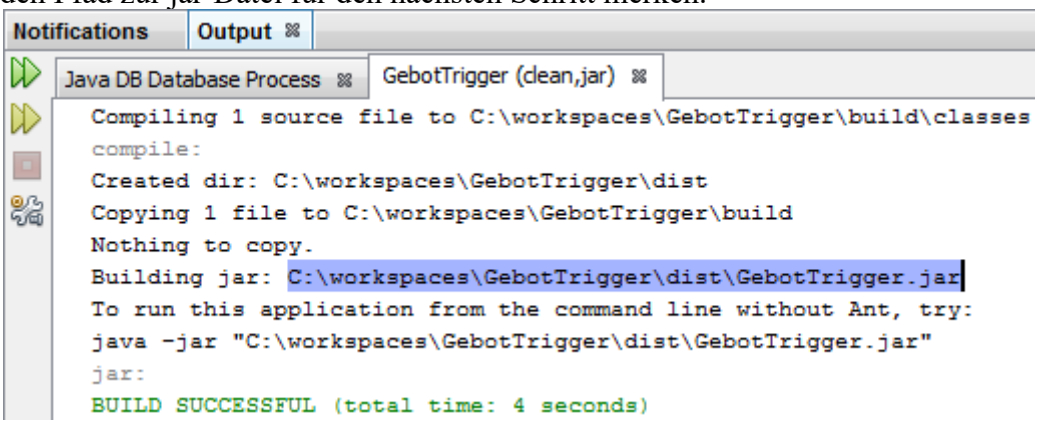
Diese Java-Methode steht in einem normalen NetBeans-Java-Projekt in einer frei gewählten Klasse, hier GebotTrigger.java im Paket gebottrigger. Das Projekt hat keine main-Methode, natürlich könnten Tests ergänzt werden. Weiterhin werden keine Bibliotheken genutzt, da nur die Standard-JDBC-Schnittstellen auftreten.



Diese Klasse wird übersetzt, dazu wird hier immer der Knopf „Clean & Build“ genutzt.

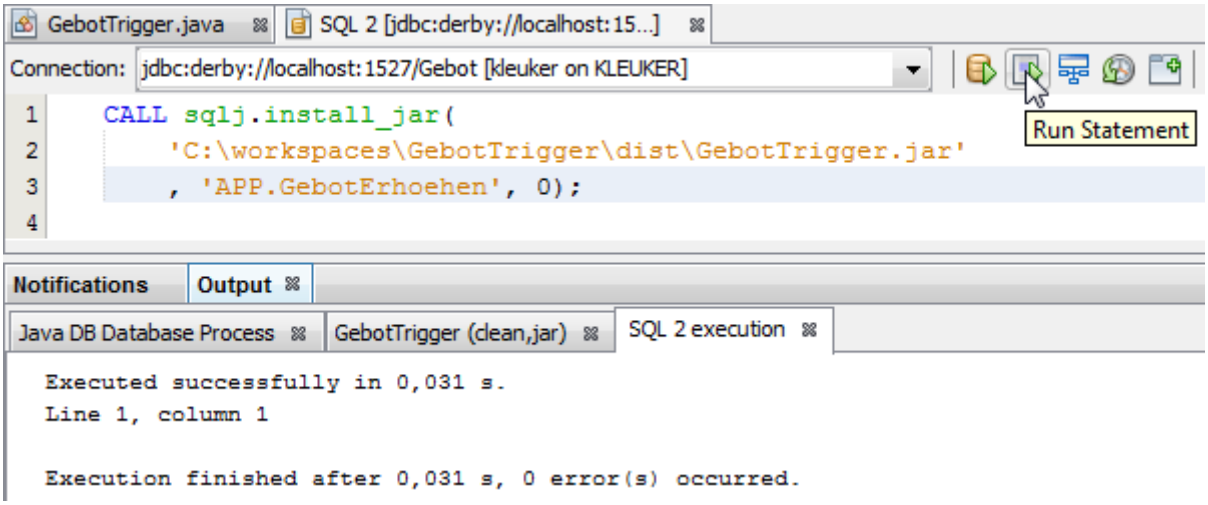


NetBeans erzeugt u.a. eine jar-Datei in die die ausführbare Klasse gepackt ist. Generell kann diese Umwandlung in eine jar-Datei natürlich auch mit anderen Werkzeugen erfolgen. In der Ausgabe muss man sich den Pfad zur jar-Datei für den nächsten Schritt merken.

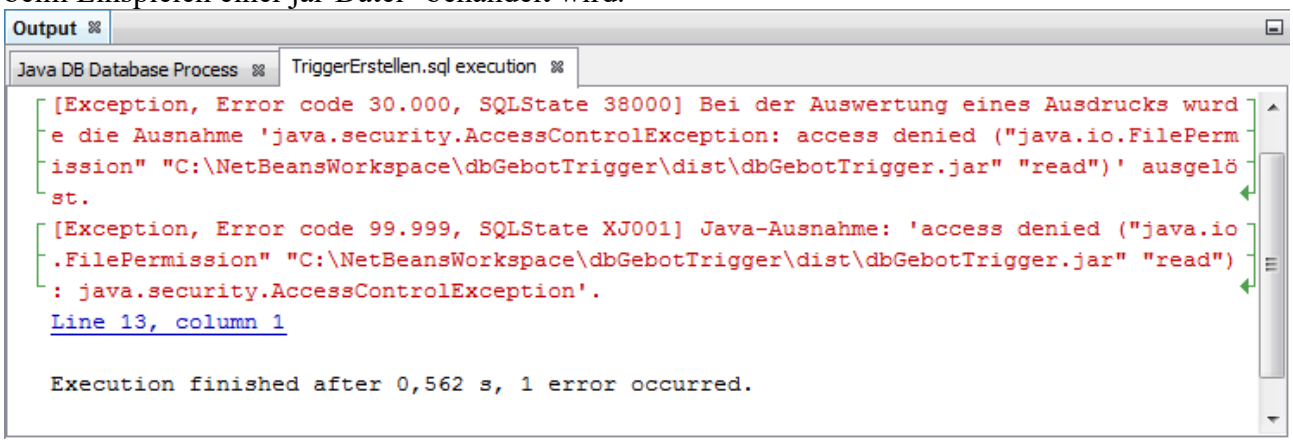


Im nächsten Schritt wird die jar-Datei in die Datenbank geladen, dazu wird ein Befehlsfenster für die Datenbank aufgemacht und der folgende Befehl eingegeben, bei dem der vorher gemerkte Pfad eine Rolle spielt. Der weiterhin angegebene Name „APP.GebotErhoehen“ ist ein relativ willkürlicher Bezeichner, der nur eindeutig in der Datenbank sein muss. Durch den Präfix „APP.“ findet eine Zuordnung zu einem Schema statt, die hier keine Rolle spielt.

```
CALL sqlj.install_jar(
    'C:\workspaces\GebotTrigger\dist\GebotTrigger.jar', 'APP.GebotErhoehen', 0);
```

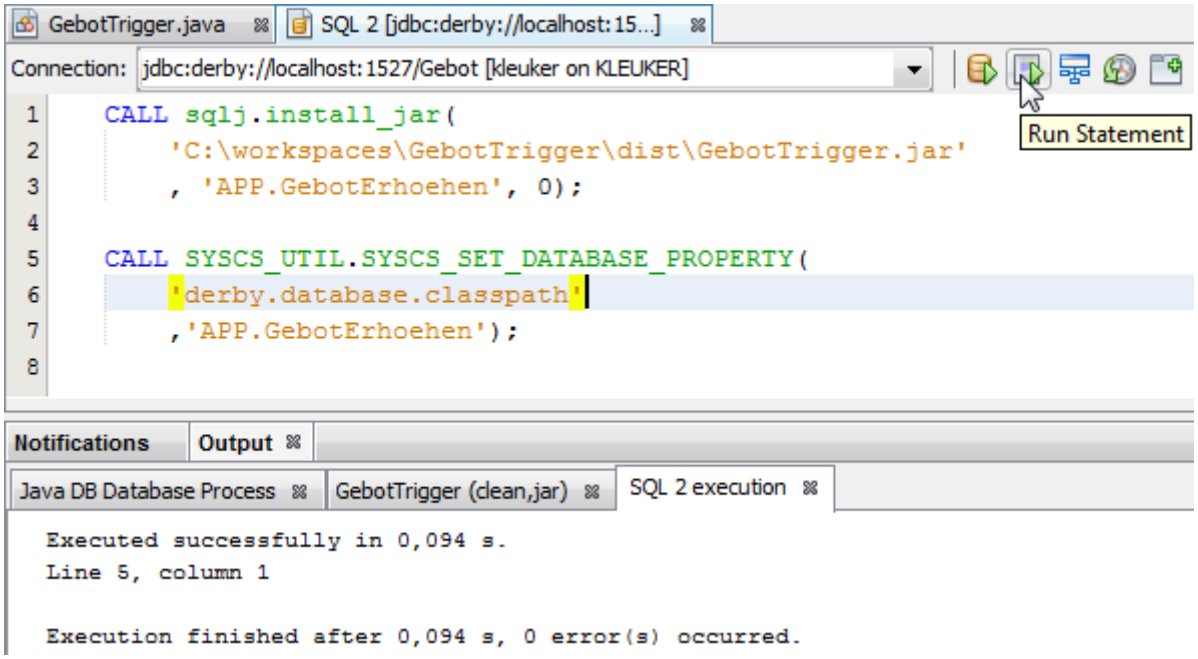


Ab der Derby-Version 10.14 wurde das Sicherheitskonzept geändert, so dass es zu folgender Fehlermeldung kommen kann, die im nachfolgenden Unterkapitel „6.9 AccessControlException beim Einspielen einer jar-Datei“ behandelt wird.



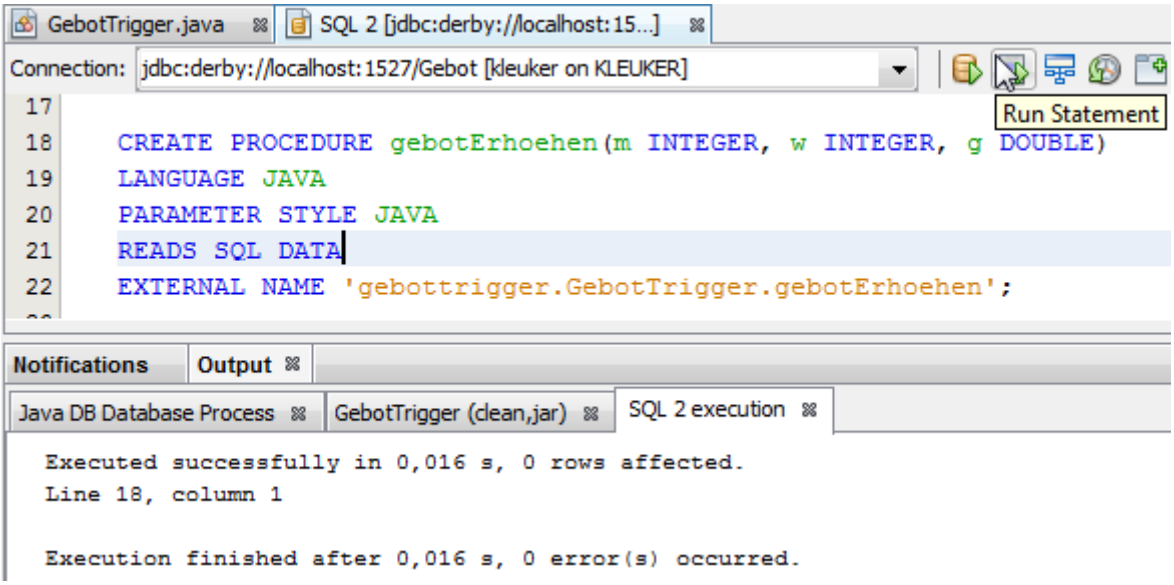
Wurde die jar-Datei erfolgreich zur Datenbank hinzugefügt, muss die Datei zu den ausführbaren Dateien der Datenbank, genauer zu deren Pfad, hinzugefügt werden, was mit dem folgenden Befehl erfolgt. Der vorher ausgewählte Name „APP.GebotErhoehen“ ist hier wieder zu nutzen. Es wird nur der letzte Befehl im Fenster ausgeführt.

```
CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY(
    'derby.database.classpath',
    'APP.Sample1');
```



Nun muss eine Prozedur angelegt werden, mit der die Java-Methode aufrufbar wird. Dabei müssen die Parameter vom Typ zu den Parametern der Java-Methode passen. Der Name der Prozedur ist frei wählbar, der externe Name ergibt sich aus dem vollqualifizierten Klassennamen, hier gebottrigger.GebotTrigger und dem Methodennamen. Die Ergänzung „READS SQL DATA“ ist wichtig und verweist darauf, dass in der Prozedur in der Datenbank gelesen, die Daten aber nicht modifiziert werden. Auf etwaige Tippfehler in Namen wird man an dieser Stelle leider nicht hingewiesen.

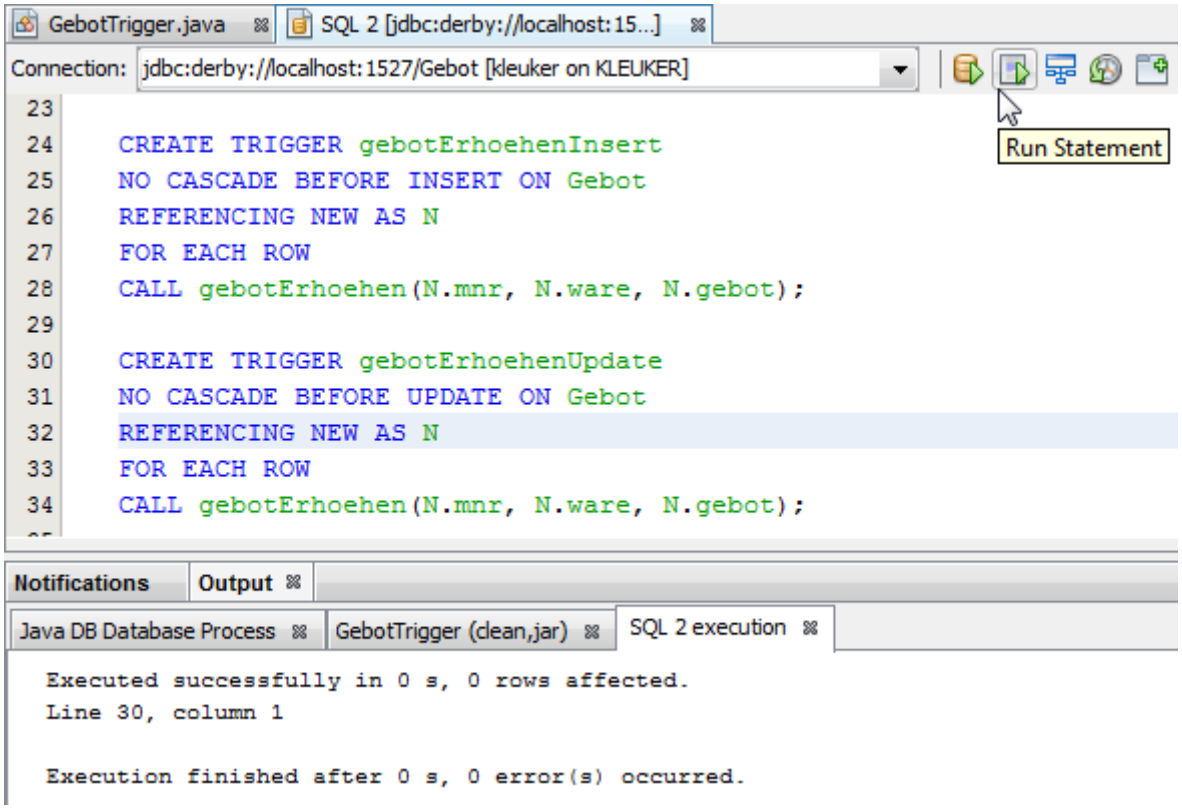
```
CREATE PROCEDURE gebotErhoehen(m INTEGER, w INTEGER, g DOUBLE)
LANGUAGE JAVA
PARAMETER STYLE JAVA
READS SQL DATA
EXTERNAL NAME 'gebottrigger.GebotTrigger.gebotErhoehen';
```



Abschließend wird der Trigger mit dem dazu passenden Event angelegt. Im konkreten Fall wird die vorher angelegte Prozedur aufgerufen.

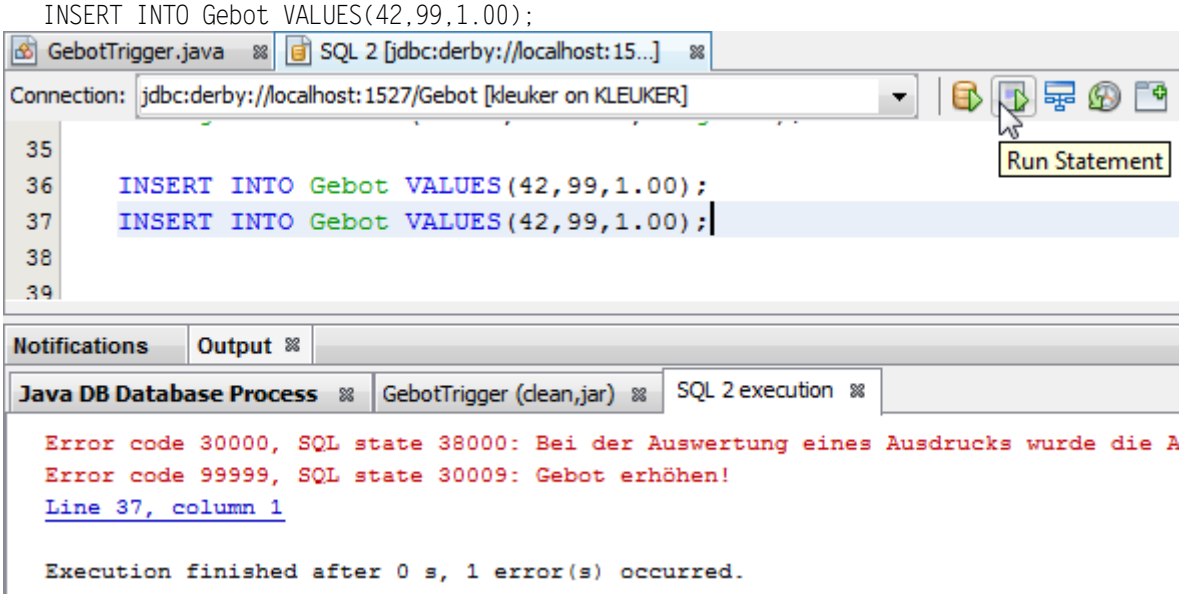
```
CREATE TRIGGER gebotErhoehenInsert
NO CASCADE BEFORE INSERT ON Gebot
REFERENCING NEW AS N
FOR EACH ROW
CALL gebotErhoehen(N.mnr, N.ware, N.gebot);

CREATE TRIGGER gebotErhoehenUpdate
NO CASCADE BEFORE UPDATE ON Gebot
REFERENCING NEW AS N
FOR EACH ROW
CALL gebotErhoehen(N.mnr, N.ware, N.gebot);
```



Danach können die Trigger ausprobiert werden. Der zweite Befehl erzeugt dabei den erwarteten Abbruch. Nebenbei wird deutlich, dass die Trigger-Prüfung vor der Constraint-Prüfung, hier dem Primary Key, erfolgt.

```
INSERT INTO Gebot VALUES(42,99,1.00);
INSERT INTO Gebot VALUES(42,99,1.00);
```

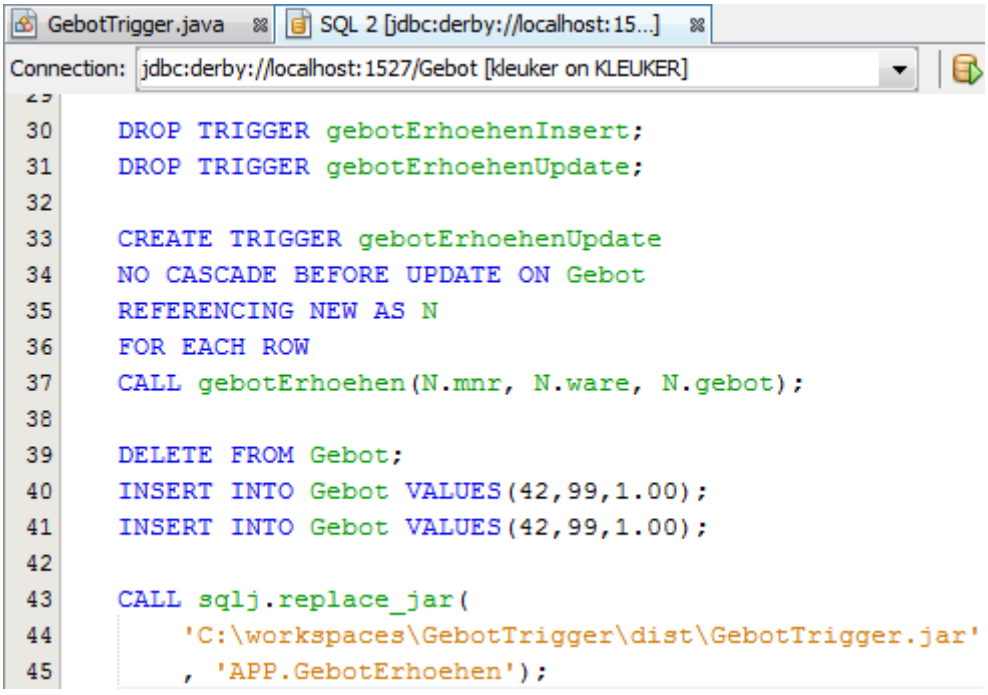


In der Realität ist es sehr unwahrscheinlich, dass die erste gewählte Lösung funktioniert, oft muss

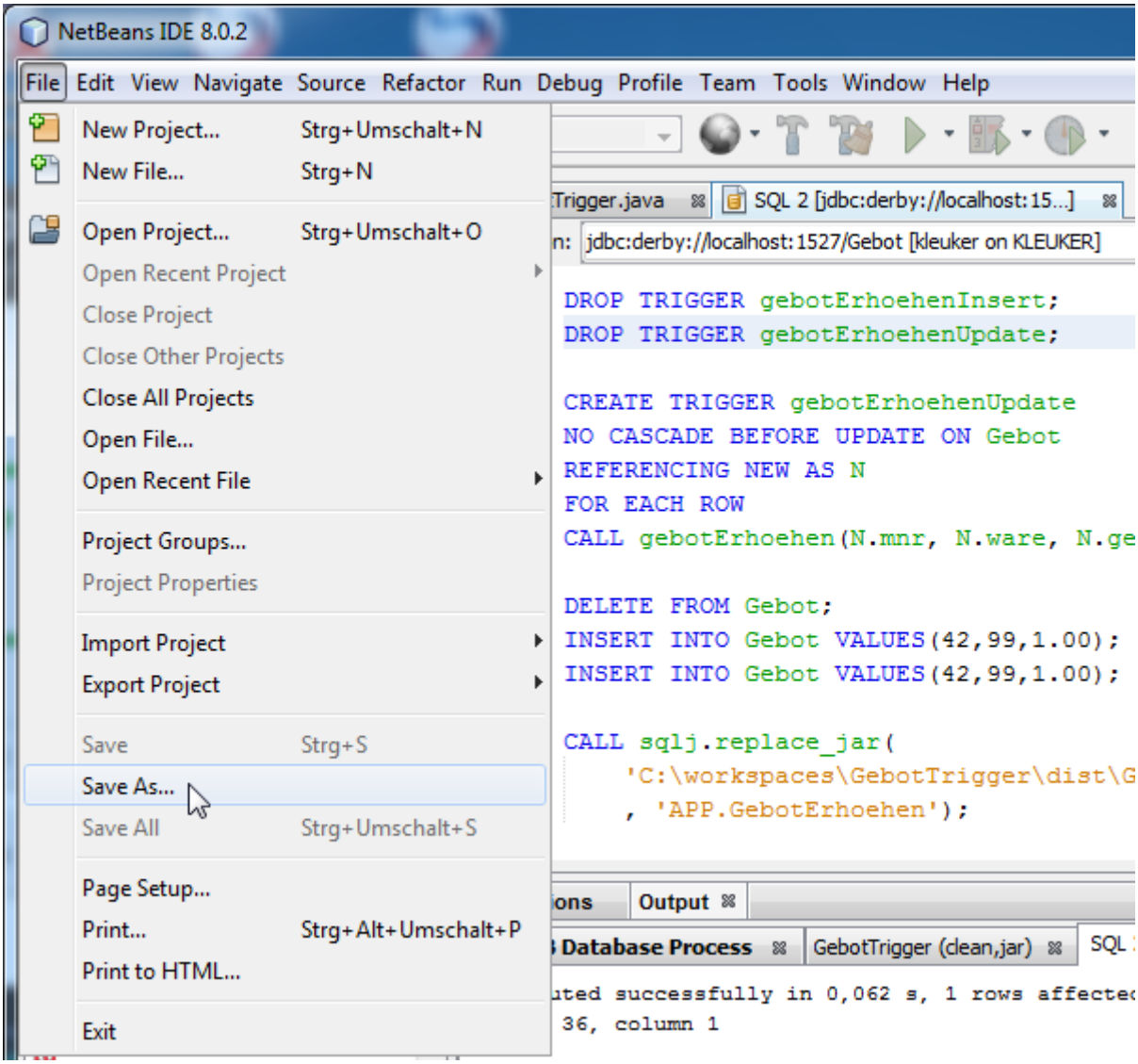
dabei das Java-Programm angepasst werden. Dazu sind nicht genau die bisher beschriebenen Schritte zu nutzen. Es sind folgende Schritte durchzuführen.

1. Änderung des Java-Programms
2. Übersetzung mit Clean & Build
3. Die jar-Datei muss ersetzt werden, dazu wird der folgende Befehl genutzt
   ```
   CALL sqlj.replace_jar(
       'C:\workspaces\GebotTrigger\dist\GebotTrigger.jar'
       , 'APP.GebotErhoehen');
   ```
4. Die anderen Schritte sind nicht zu wiederholen, der Trigger kann direkt neu getestet werden, wobei eventuell die vorher eingefügten Daten zu löschen sind.

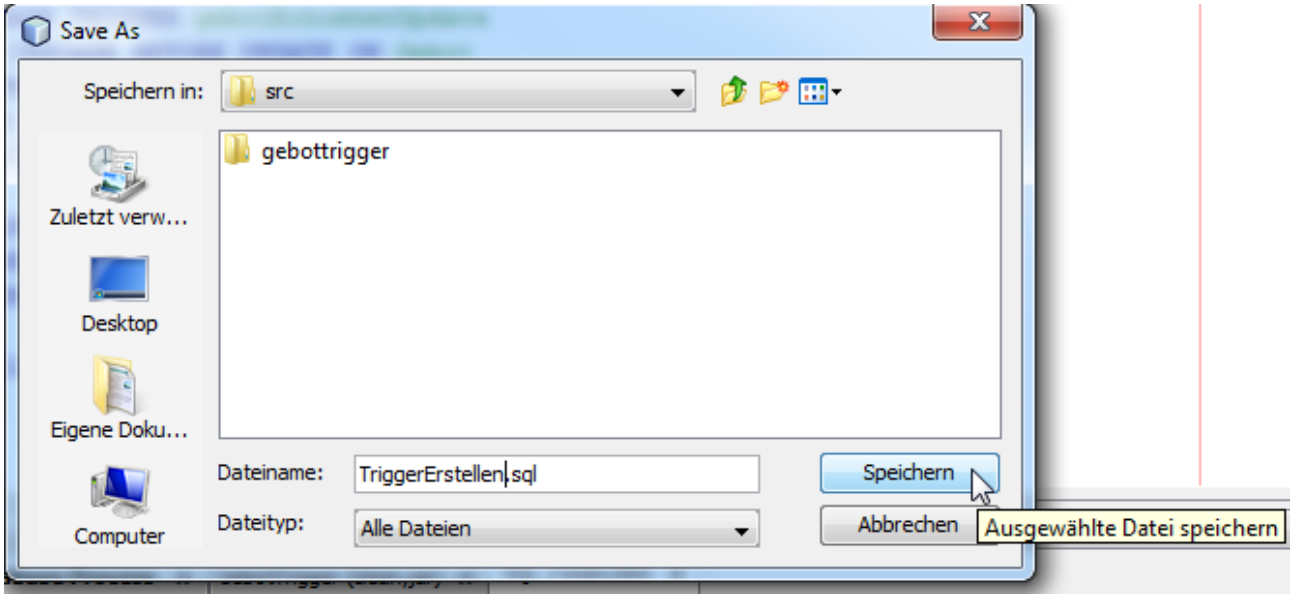
Generell sollte während der Arbeit einfach ein Arbeitsblatt mit allen wichtigen Datenbankbefehlen in einem Reiter jederzeit verfügbar sein, so dass die Befehle schnell ausführbar sind. Will man eine Prozedur oder einen Trigger in der Struktur verändern, müssen diese vorher mit einem DROP-Befehl gelöscht und dann wieder angelegt werden. Der Ausschnitt aus einem Beispielblatt kann wie folgt aussehen.



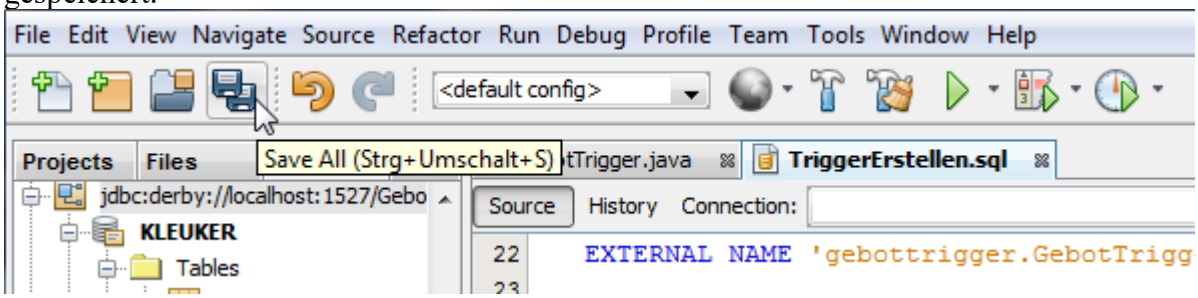
Solche Arbeitsblätter werden nicht in den NetBeans-Projekten gespeichert, deshalb sollte man dies von Hand an einer beliebigen Stelle im Projekt tun. Bei geöffneten Arbeitsblatt wird „File -> Save As…“ gewählt.



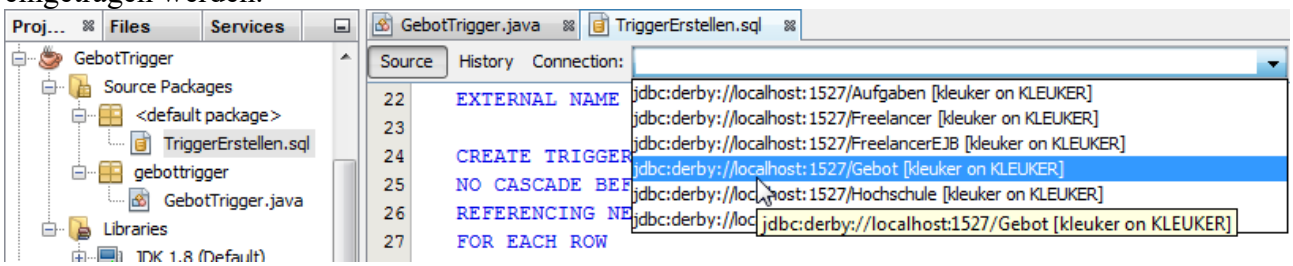
Es wird z. B. in das src-Verzeichnis des Projekts gesteuert und die Datei mit einer Endung „.sql“ abgespeichert.



Danach wird die SQL-Datei auch zusammen mit dem Projekt und dem Klick auf „Save all“ gespeichert.



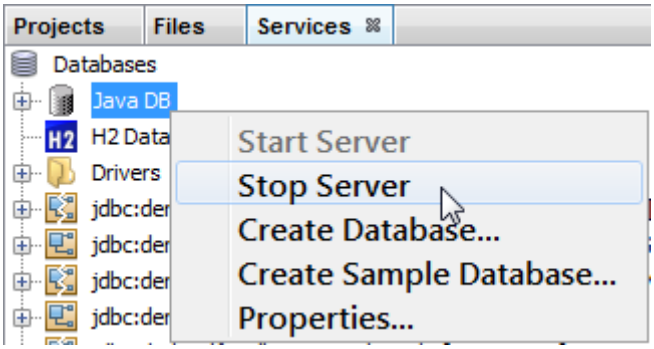
Wird die Datei geschlossen und später wieder geöffnet, muss zunächst oben die Connection eingetragen werden.



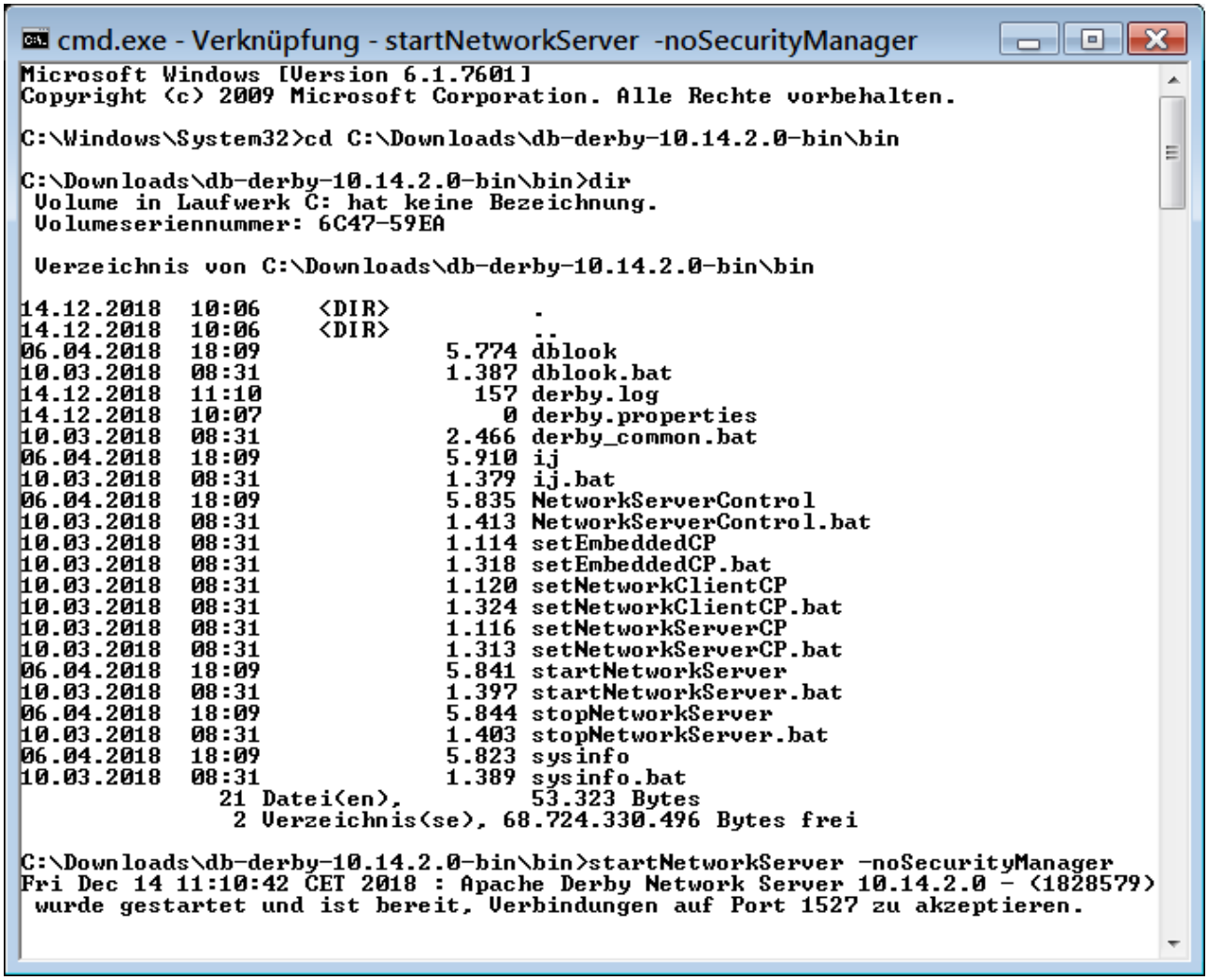
6.9 AccessControlException beim Einspielen einer jar-Datei

Ab der Version 10.14 hat Derby das Sicherheitskonzept geändert und enger mit den Java-Security-Einstellungen verknüpft. Dies betrifft u. a. die Möglichkeiten zusätzliche Software in Form von jar-Dateien in die Datenbank einzuspielen. Generell ist dieser Schritt sinnvoll, erschwert aber das Experimentieren mit Triggern. Ohne detaillierte Betrachtung der Java-Security-Mechanismen, meist

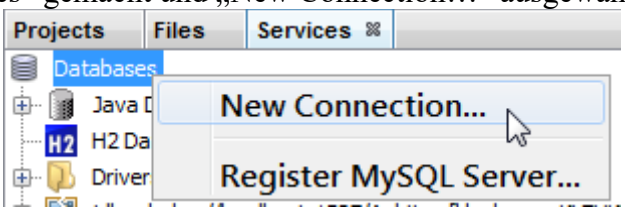
basierend auf der Datei \Java\<jdk_version>\jre\lib\security\java.security, kann man entweder zu einer früheren Version von Derby wechseln, wobei dann die benutzten Datenbanken neu zu erstellen sind oder den folgenden Workaround nutzen.
Zunächst darf die die Datenbank nicht laufen, sie muss gegebenenfalls über einen Rechtsklick und „Stop Server“ angehalten werden.



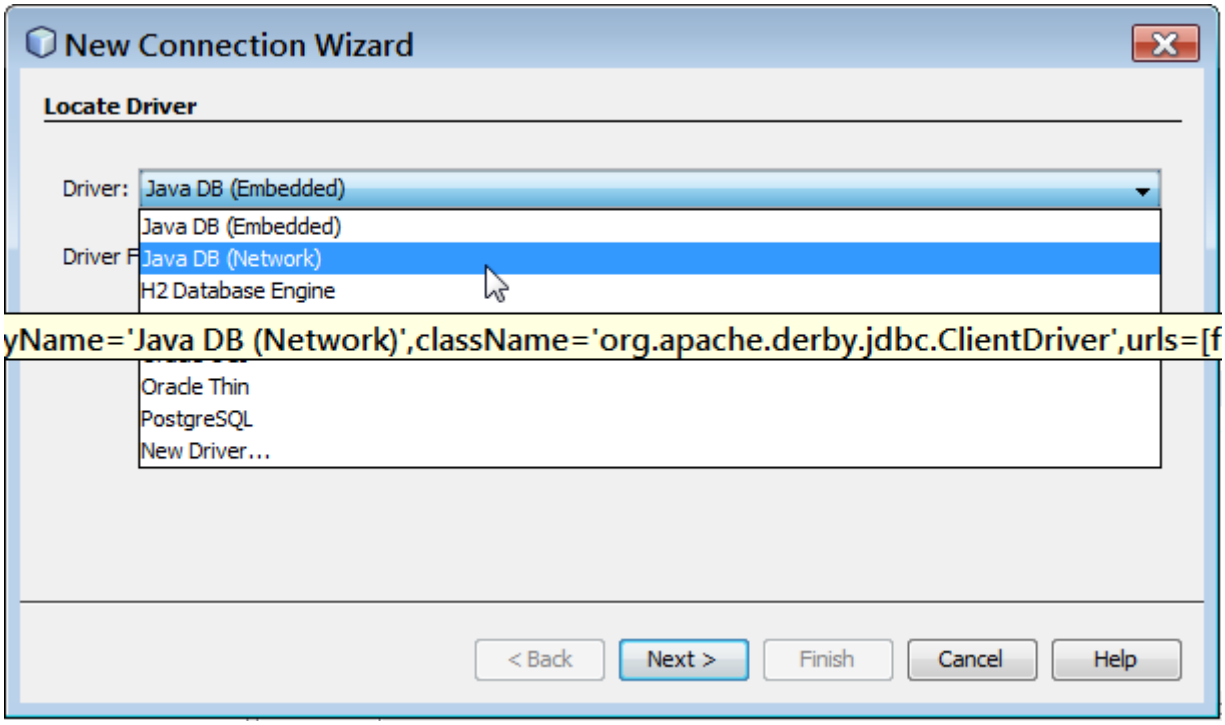
Danach wird in einer cmd-Shell in den Installationsordner der Datenbank gewechselt und die über den Befehl `startNetworkServer -noSecurityManager` gestartet. Es sollte klar sein, dass diese Einstellung nur zum Experimentieren sinnvoll ist.



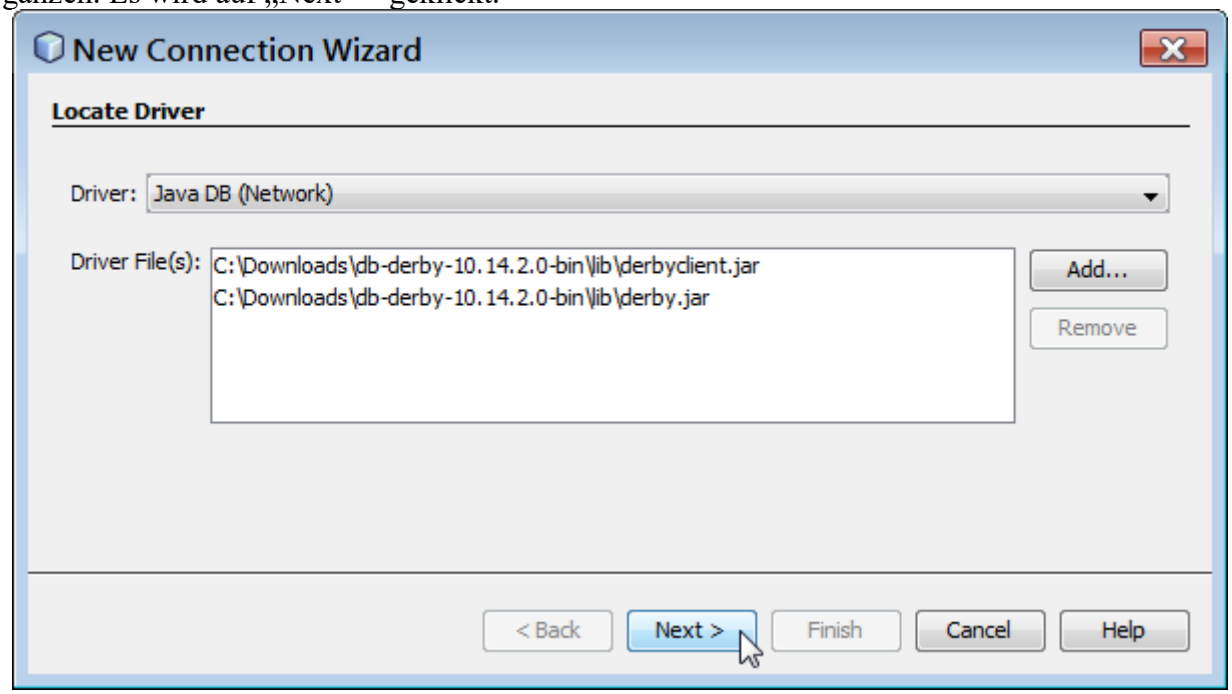
Nun muss in NetBeans manuell eine Verbindung zur Datenbank erstellt werden. Datz wird ein Rechtsklick auf „Databases“ gemacht und „New Connection…“ ausgewählt.



Unter „Driver“ wird „Java DB (Network)“ ausgewählt.

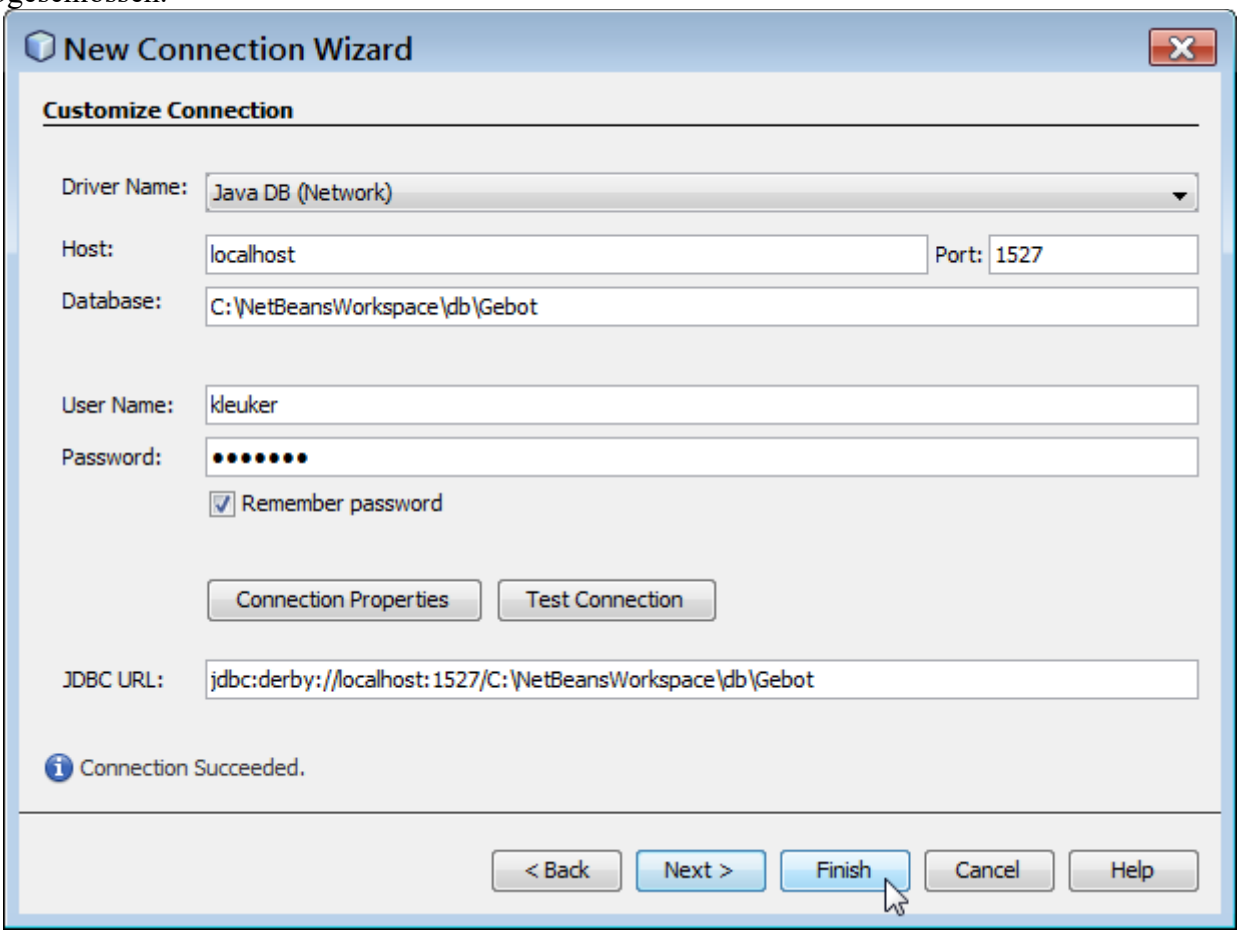


Es erscheinen automatisch die passenden „Driver Files(s)“. Falls nicht, sind dies über „Add…“ zu ergänzen. Es wird auf „Next >“ geklickt.

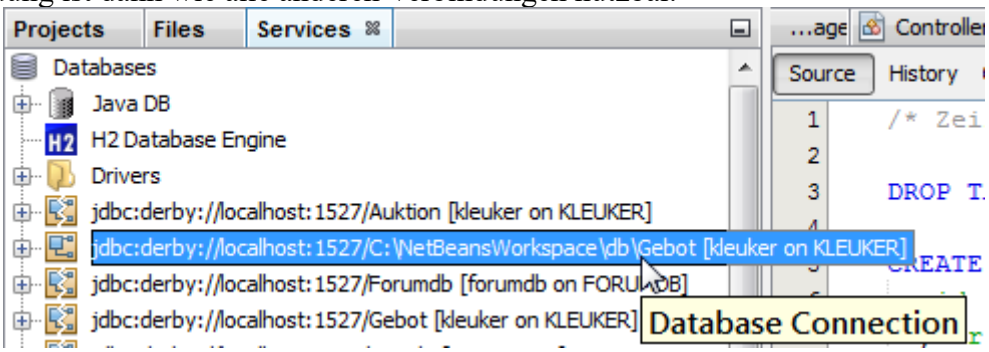


Es ist sinnvoll zunächst unten die „JDBC URL“ einzutragen, da dann einige Einträge oben automatisch gefüllt werden, „User Name“ und „Passwort“ sind anzugeben. Im Eintrag

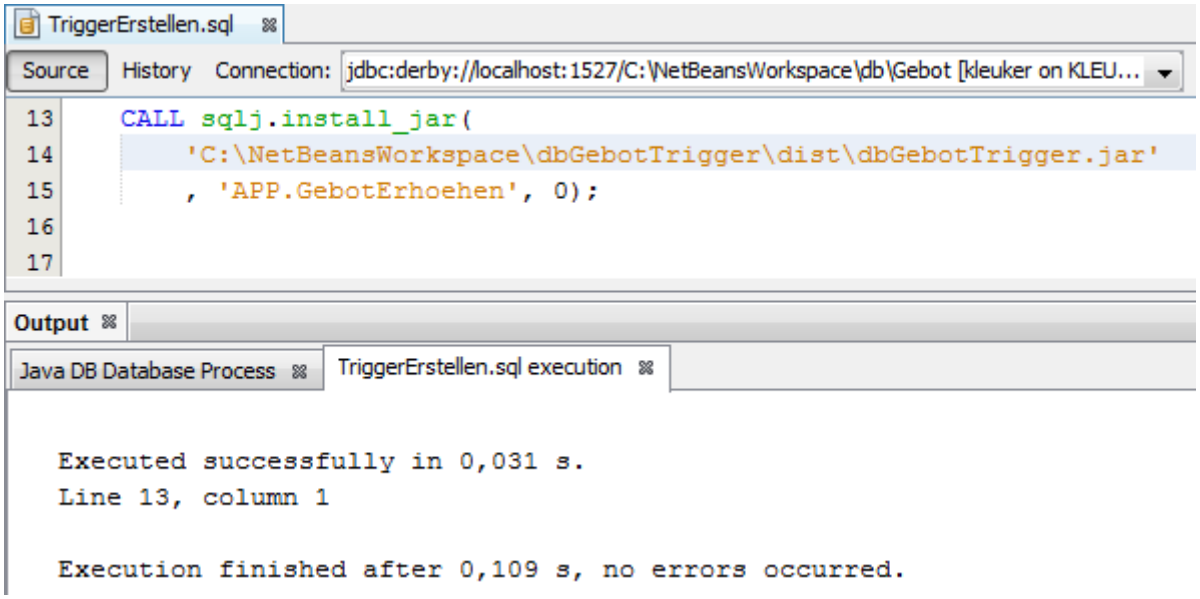
`jdbc:derby://localhost:1527/C:\NetBeansWorkspace\db\Gebot` ist der Pfad zur Datenbank zu erkennen, der beim Datenbanknamen ergänzt wurde. Über den Knopf „Test Connection“ wird der Verbindungsaufbau überprüft. Danach wird die Erstellung der Verbindung über „Finish“ abgeschlossen.



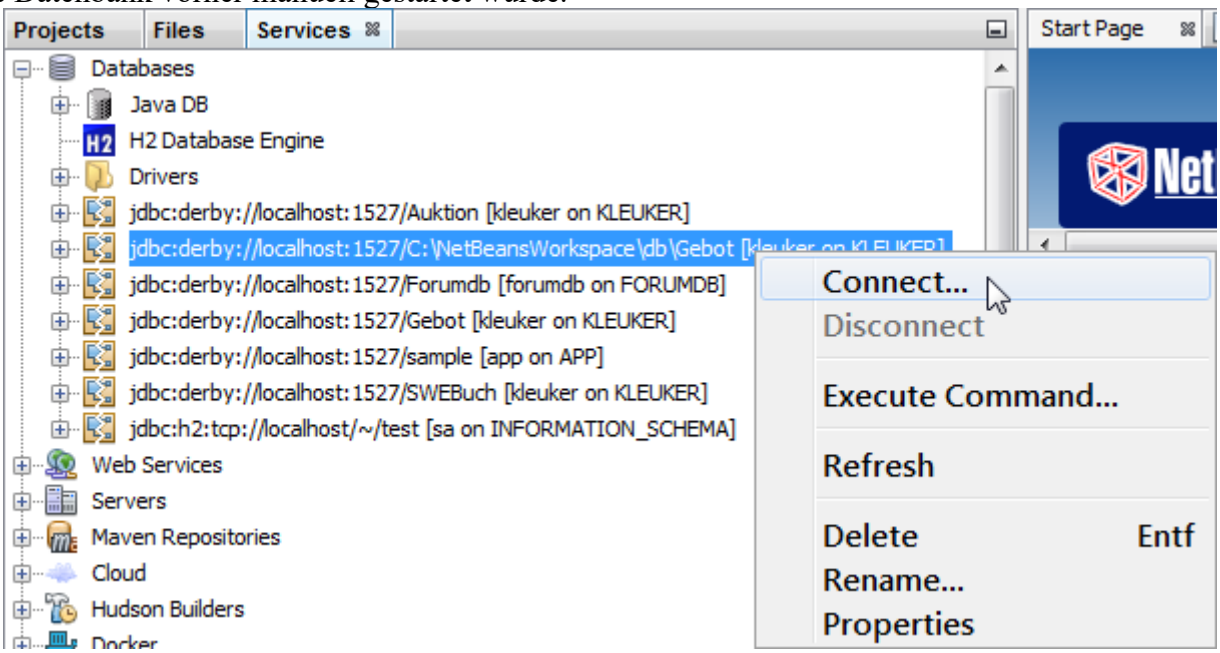
Die Verbindung ist dann wie alle anderen Verbindungen nutzbar.



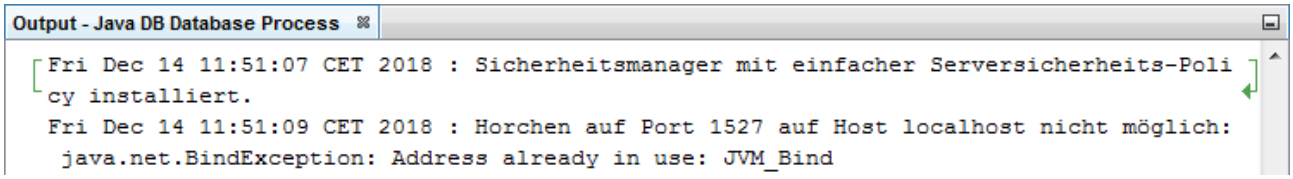
Das Einspielen von jar-Dateien ist problemlos möglich.



Die Verbindung ist natürlich auch nach einem Neustart von NetBeans nutzbar. Wichtig ist nur, dass die Datenbank vorher manuell gestartet wurde.

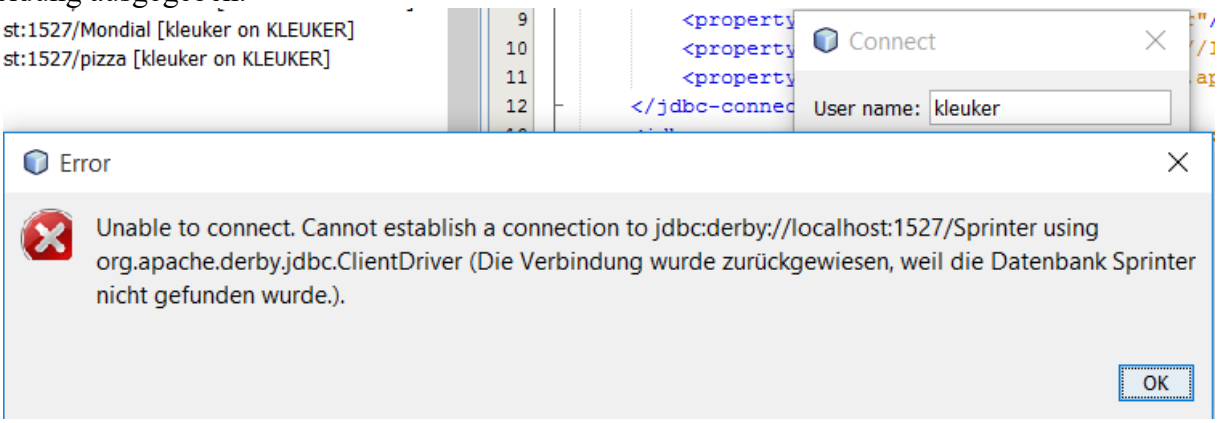


Da Derby-Verbindungen generell von NetBeans verwaltet werden, versucht NetBeans zusätzlich die Datenbank zu starten. Dies scheitert mit der folgenden Meldung. Da die Datenbank bereits von Hand gestartet wurde, ist dies kein Problem und die Meldung kann ignoriert werden. Sollte die Datenbank von NetBeans gestartet werden, würden die Security-Probleme wieder auftreten.

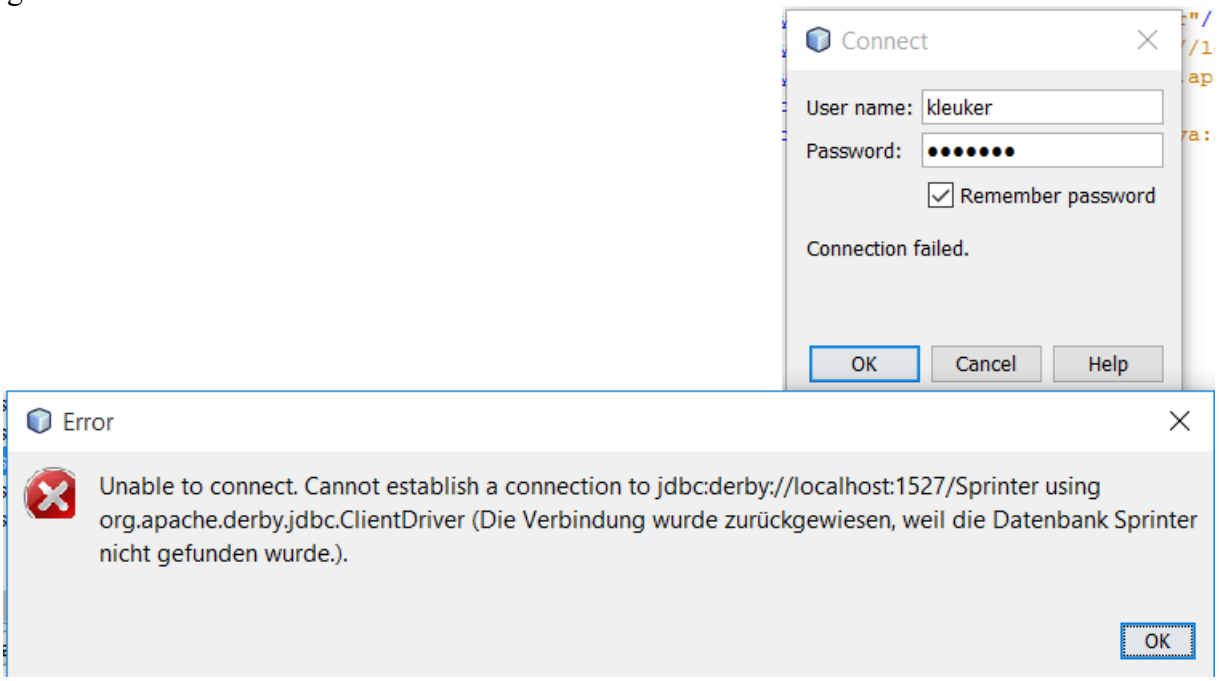


6.10 Zugriffsprobleme durch verschiedene Betriebssysteme oder Nutzer

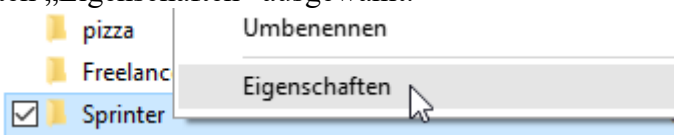
Bei der Nutzung mehrerer Rechner, verschiedener Nutzer oder dem Austausch des Betriebssystems kann es zu Problemen mit den Rechten auf Dateien, gerade bei gesicherten Datenbank-Dateien, kommen. Wird versucht so eine Datenbank mit einem Doppelklick zu öffnen, wird die folgende Meldung ausgegeben.



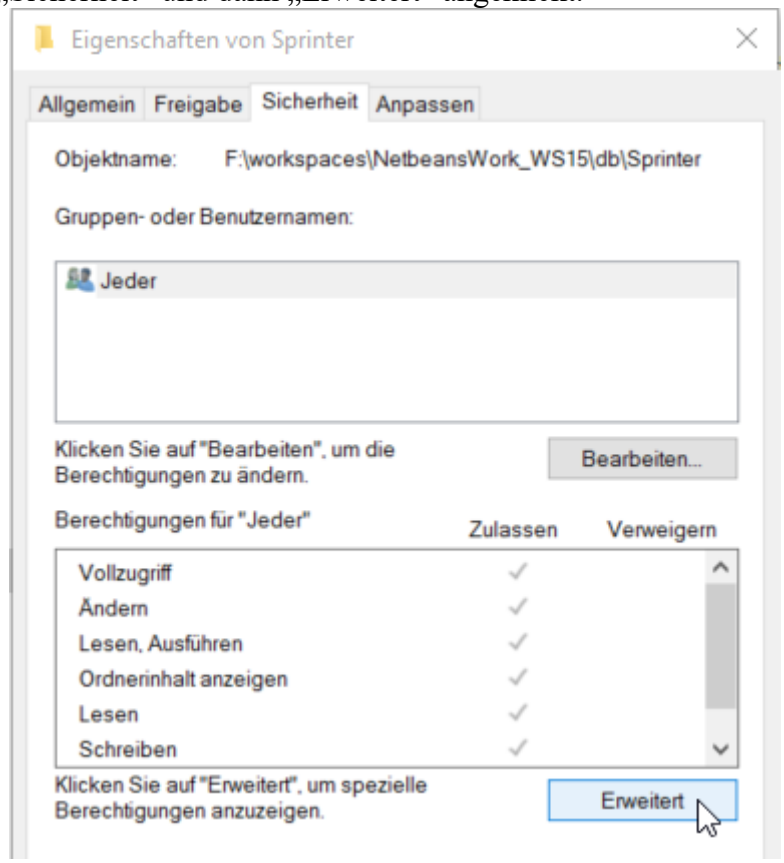
Nach einem Klick auf „Ok“ kann zwar das Passwort eingegeben werden, die nächste Ausgabe zeigt aber, dass die Datenbank nicht gefunden wurde. Dies wird auch ausgegeben, wenn der Zugriff nicht möglich ist und die Datenbank existiert.



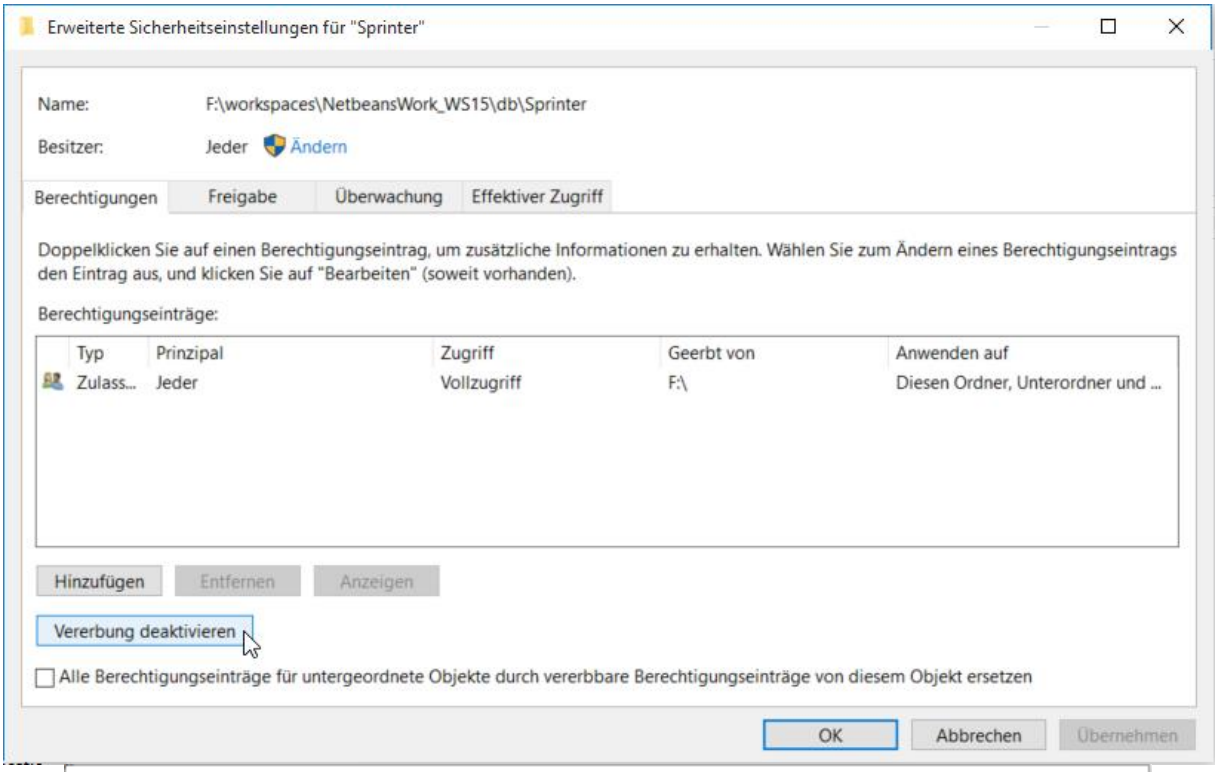
Es müssen die Rechte an dem Datenbankordner geändert werden. Dazu wird ein Rechtsklick auf dem Ordner gemacht und unten „Eigenschaften“ ausgewählt.

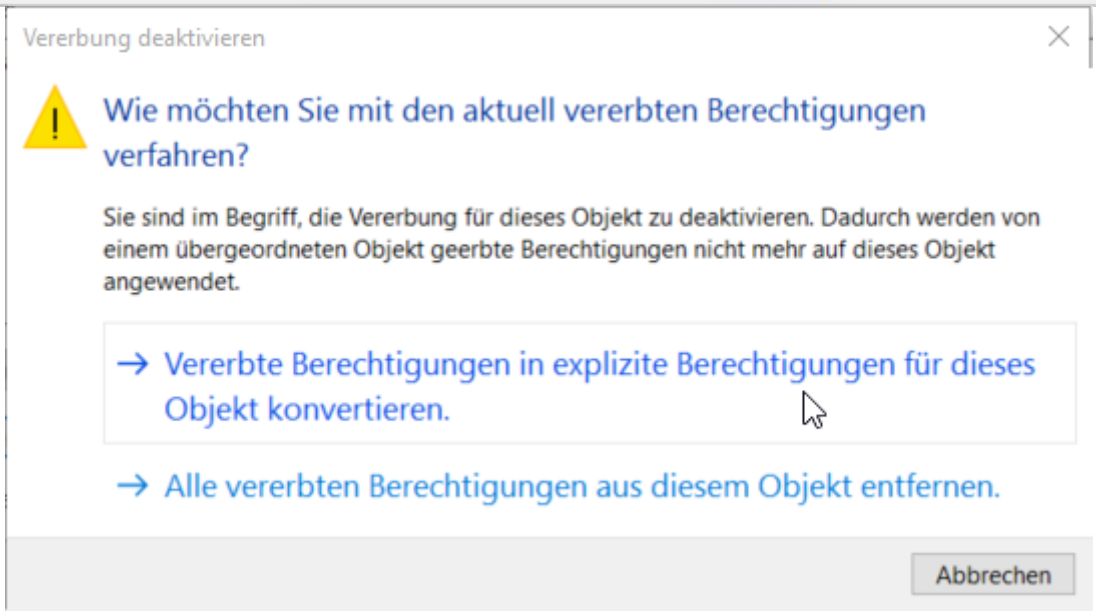


Es wird der Reiter „Sicherheit“ und dann „Erweitert“ angeklickt.

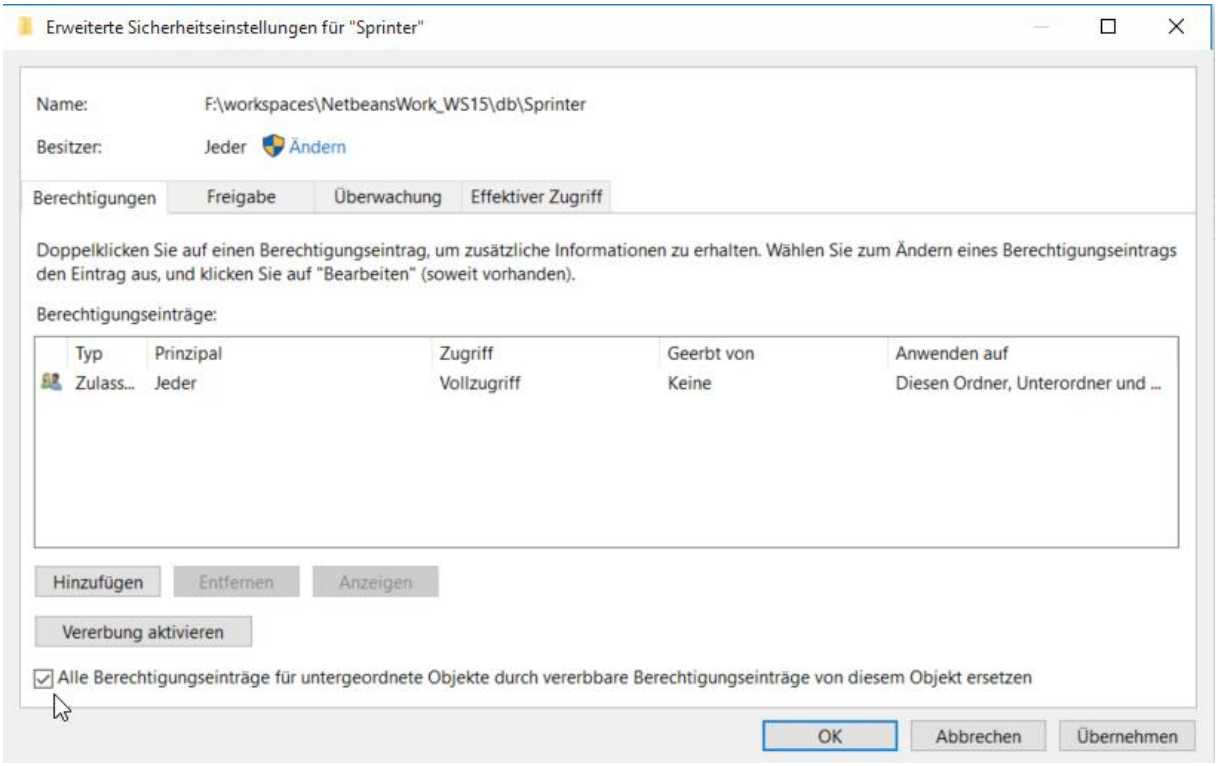


Zunächst wird ein Klick auf „Vererbung deaktivieren“ gemacht und danach der erste Eintrag mit „konvertieren“ gewählt.

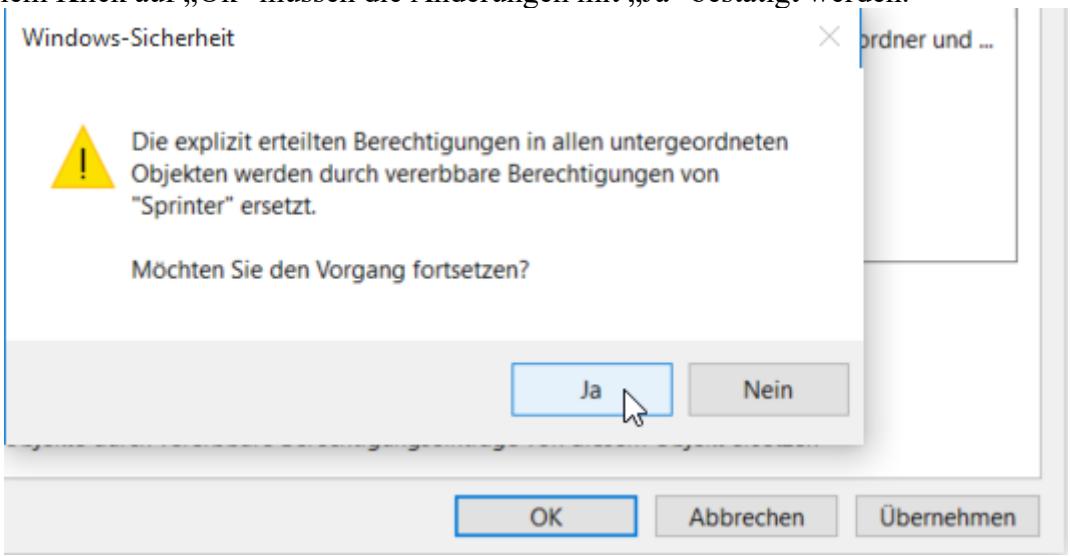




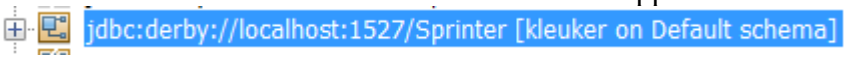
Nun wird unten der Haken für „Alle Berechtigungseinträge…“ gesetzt.



Nach einem Klick auf „Ok“ müssen die Änderungen mit „Ja“ bestätigt werden.



Danach sollte die Datenbank z. B. durch einen Doppelklick ohne Probleme geöffnet werden können.

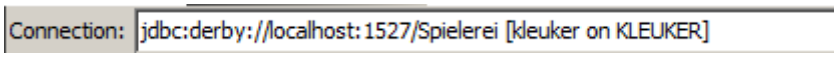


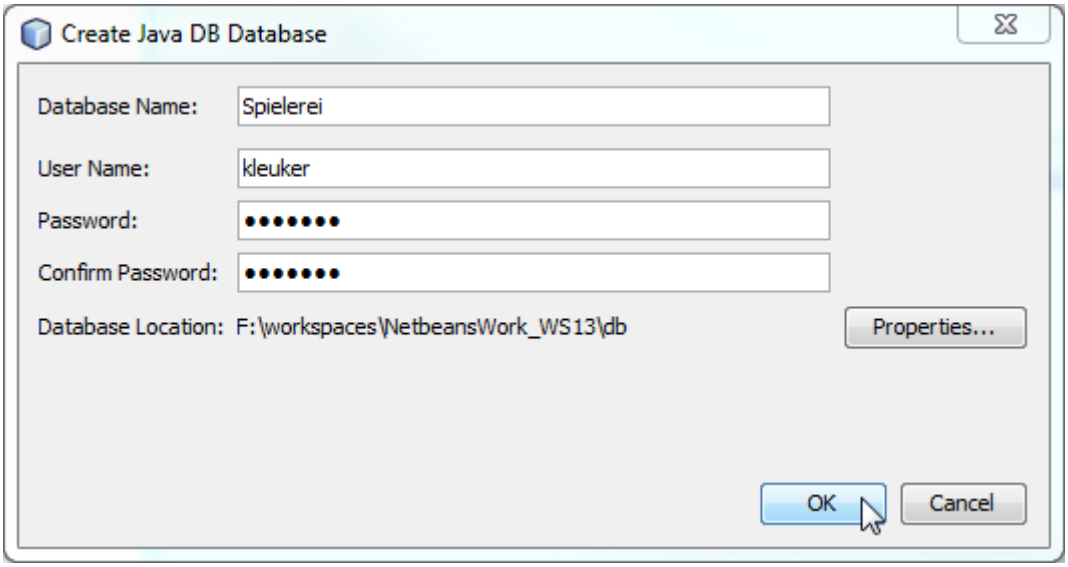
7 JPA (NetBeans 8)

In diesem Abschnitt wird gezeigt, wie man die Java Persistence API zur Verwaltung von Java-Objekten nutzen kann.

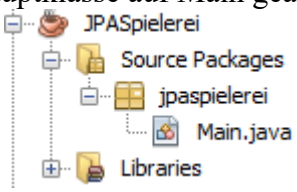
7.1 Vorbereitung der JPA-Nutzung

Es wird davon ausgegangen, dass eine Java DB-Datenbank mit Namen „Spielerei“, Nutzer und Passwort „kleuker“ existiert. Generell muss es einen Nutzer mit einem nicht leeren Passwort geben.

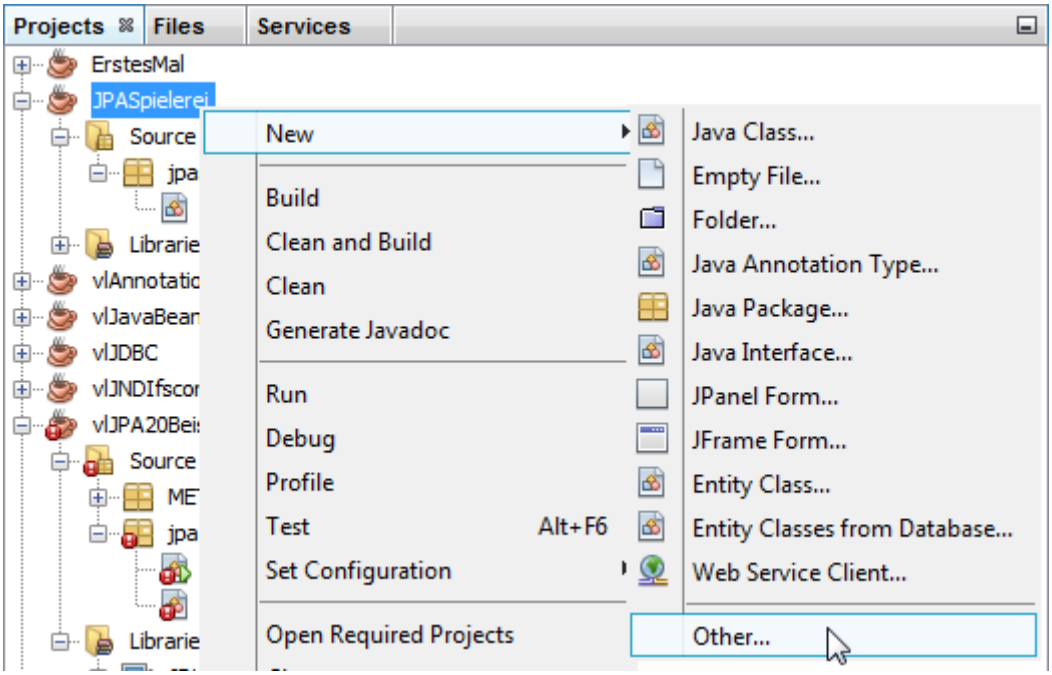




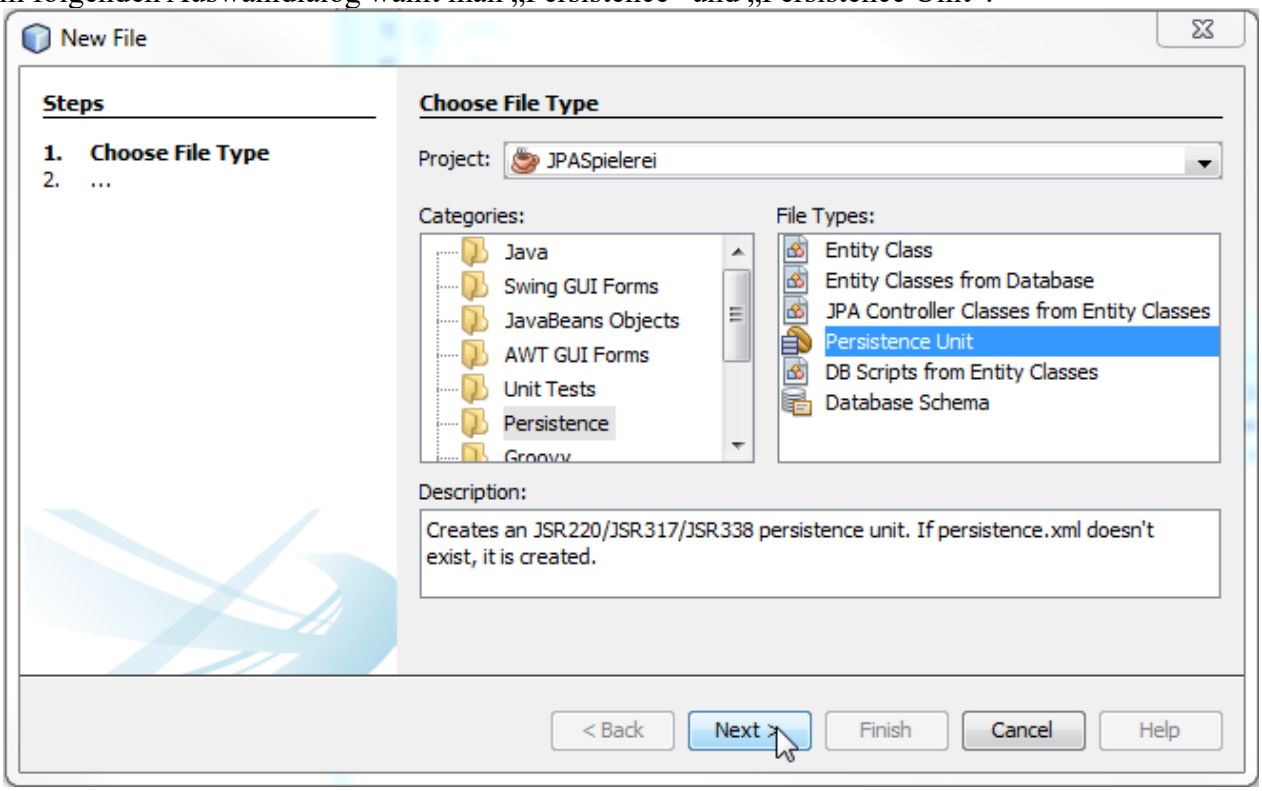
Die Datenbank enthält noch keine Tabellen.
Weiterhin wird ein normales Java-Projekt namens JPASpielerei mit den Standardeinstellungen angelegt, dabei wird der Name der Hauptklasse auf Main geändert.



Da das Erstellen der Hilfsdateien und Verzeichnisse, die z. B. für die Konfigurationsdatei persitence.xml benötigt werden, von Hand recht aufwändig ist, wird hier der Wizzard von NetBeans genutzt. Alternativ legt man zunächst selbst einfach Bean-Klassen an, annotiert diese als Entity und fügt dann die im Folgenden beschriebene Persistence Unit nachträglich zum Projekt hinzu.
Das Anlegen einer Persistence-Unit ist über einen Rechtsklick auf dem Projekt und „Neu“ und „Anderes“ (New -> Other...) möglich. Da sich die Auswahl bei „Neu“ an die letzten Projektarbeiten anpasst, könnte man im Beispiel bereits hier „Persistenzeinheit...“ (Persistency Unit) wählen.

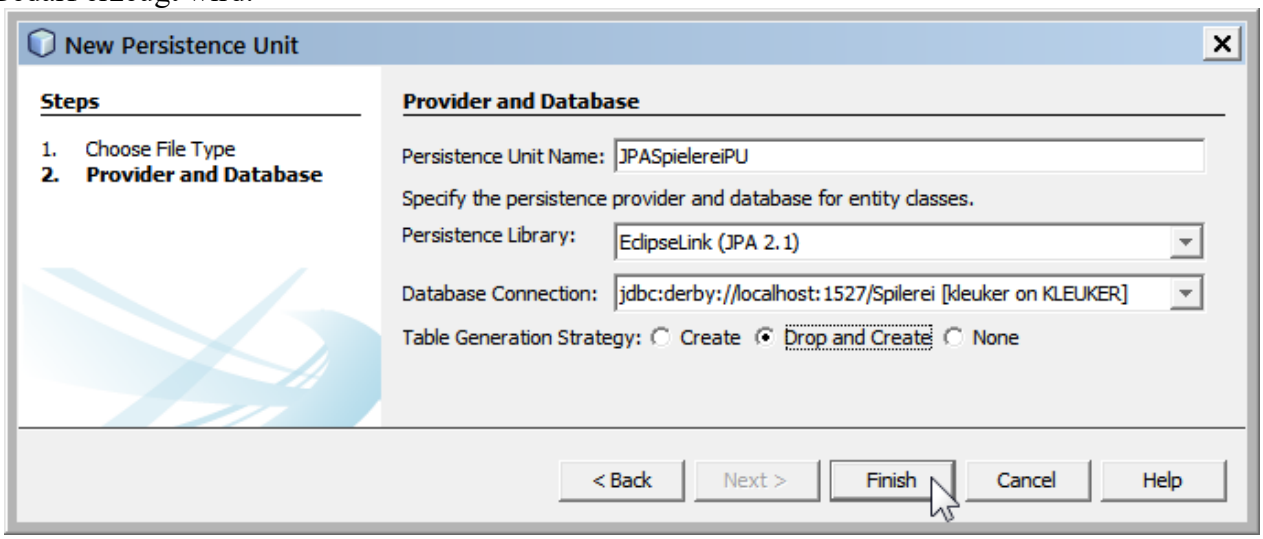


Im folgenden Auswahldialog wählt man „Persistence“ und „Persistence Unit“.

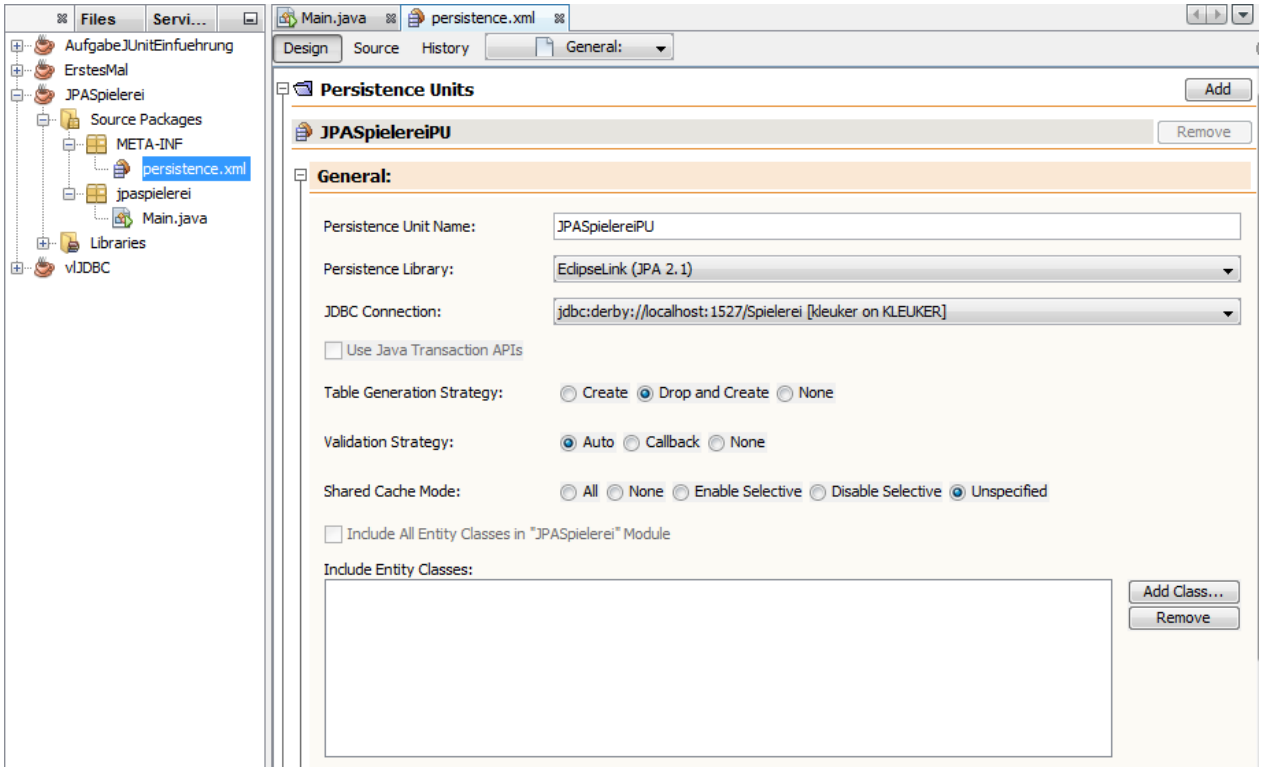


Im nachfolgenden Fenster kann die JPA-Realisierung ausgewählt und der Persistenz-Einheit ein Name gegeben werden. Unter Database Connection wird die zur Persistierung genutzte Datenbank ausgewählt, die Table Generation Strategy auf „Drop and Create“ gesetzt und „Finish“ geklickt. Mit

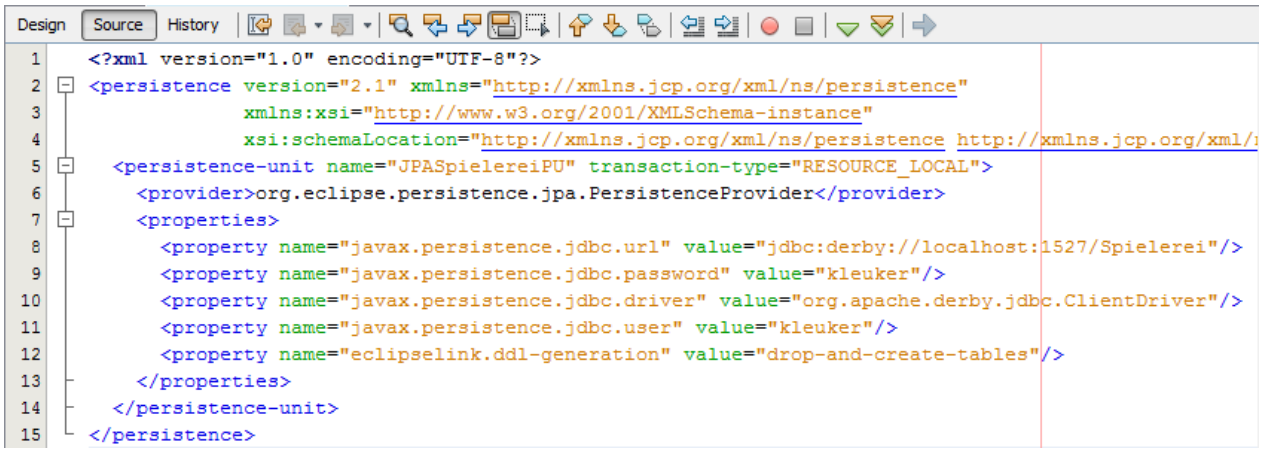
der Strategy wird der bei der Entwicklung sinnvolle Ansatz genutzt, dass bei jedem Programmaufruf die benötigten Tabellen zunächst gelöscht und wieder neu angelegt werden. Im realen Betrieb würde man „None“ wählen, da man davon ausgeht, dass eine existierende Datenbank genutzt wird. Nur zur einmaligen Erzeugung könnte man in der Praxis einmal „Create“ wählen, damit die Datenbank bei Bedarf erzeugt wird.



Die Datei persistence.xml liegt im Ordner META-INF, wo man sie auch manuell anlegen könnte. Auf der rechten Seite öffnet sich ein Editor dür die Persistence-Unit, in der im graphischern Editor süäter einige Einstellungen gemacht werden können. Generell ist es sehr sinnvoll weiter unten, den Haken bei „Include All Entity Classes in JPASpielerei Module“ zu setzen, da dann Objekte aller mit @Entity annotierten Klassen persistiert werden. Dieser Haken ist leider bei einem leeren Projekt nicht setzbar. Alternativ wird über den Knopf „Add Class…“ festgelegt, für welche mit @Entity annotierten Klassen die Persistierung der zugehörigen Objekte erfolgen soll.



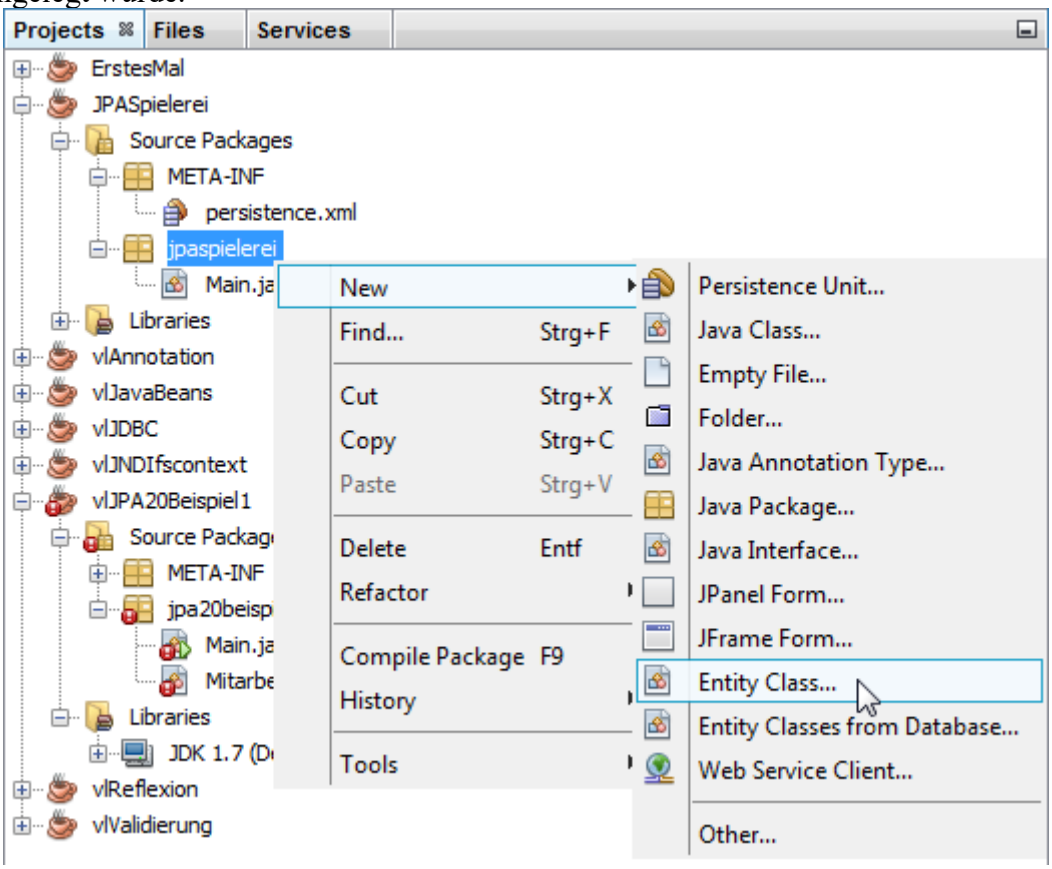
Klickt man oben auf Source, wird die Datei im XML-Format angezeigt, in dem sie ebenfalls editiert werden kann.



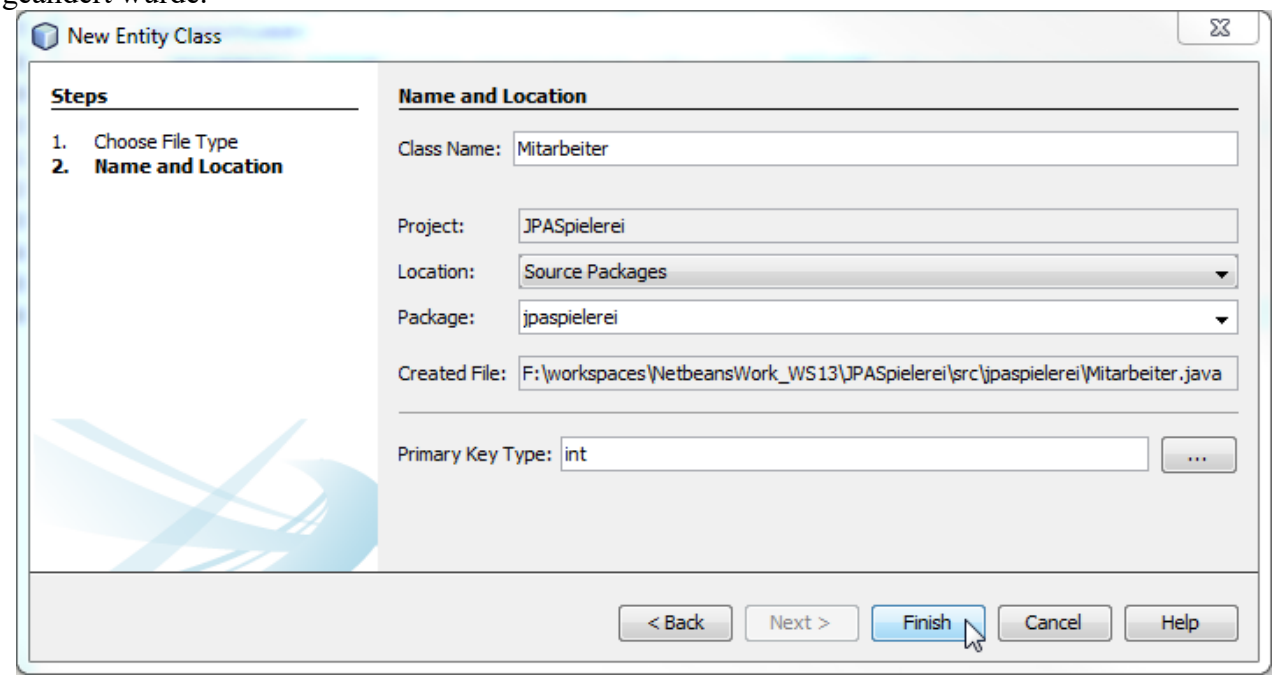
7.2 Neue Entitys anlegen

Es soll eine einfache Mitarbeiterklasse direkt mit dem Wizzard als Beispiel angelegt werden. Vorweg sei angemerkt, dass hier eine Entwicklung ohne Wizzard ähnlich schnell ist.
Am Anfang wird ein Rechtsklick auf dem Paket gemacht in dem die Entity-Klasse angelegt werden soll und „New“ und „Entity Class...“ ausgewählt. Sollte „Entity Class...“ nicht direkt wählbar sein, muss unter „Other...“ dann „Persistency“ und „Entity Class“ (eingedeutscht als „Begriffsklasse“) gewählt werden. Man erkennt in der Abbildung, dass die Datei persistence.xml im gewünschten

Ordner angelegt wurde.



Der Name der Klasse wird eingegeben und der Typ des Schlüssels ist anzugeben, der hier auf int geändert wurde.



Der folgende Programmcode wird generiert, der gut als Grundlage für die weitere Entwicklung genutzt werden kann. Für den nachfolgenden Abschnitt wurde manuell eine Exemplarvariable name mit get- und set-Methoden ergänzt.

```
package jpaspielerei;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Mitarbeiter implements Serializable {
  private static final long serialVersionUID = 1L;
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private int id;

  private String name;  // hinzu

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getName() {  // hinzu
    return name;
  }

  public void setName(String name) { // hinzu
    this.name = name;
  }

  @Override
  public int hashCode() {
    int hash = 0;
    hash += (int) id;
    return hash;
  }

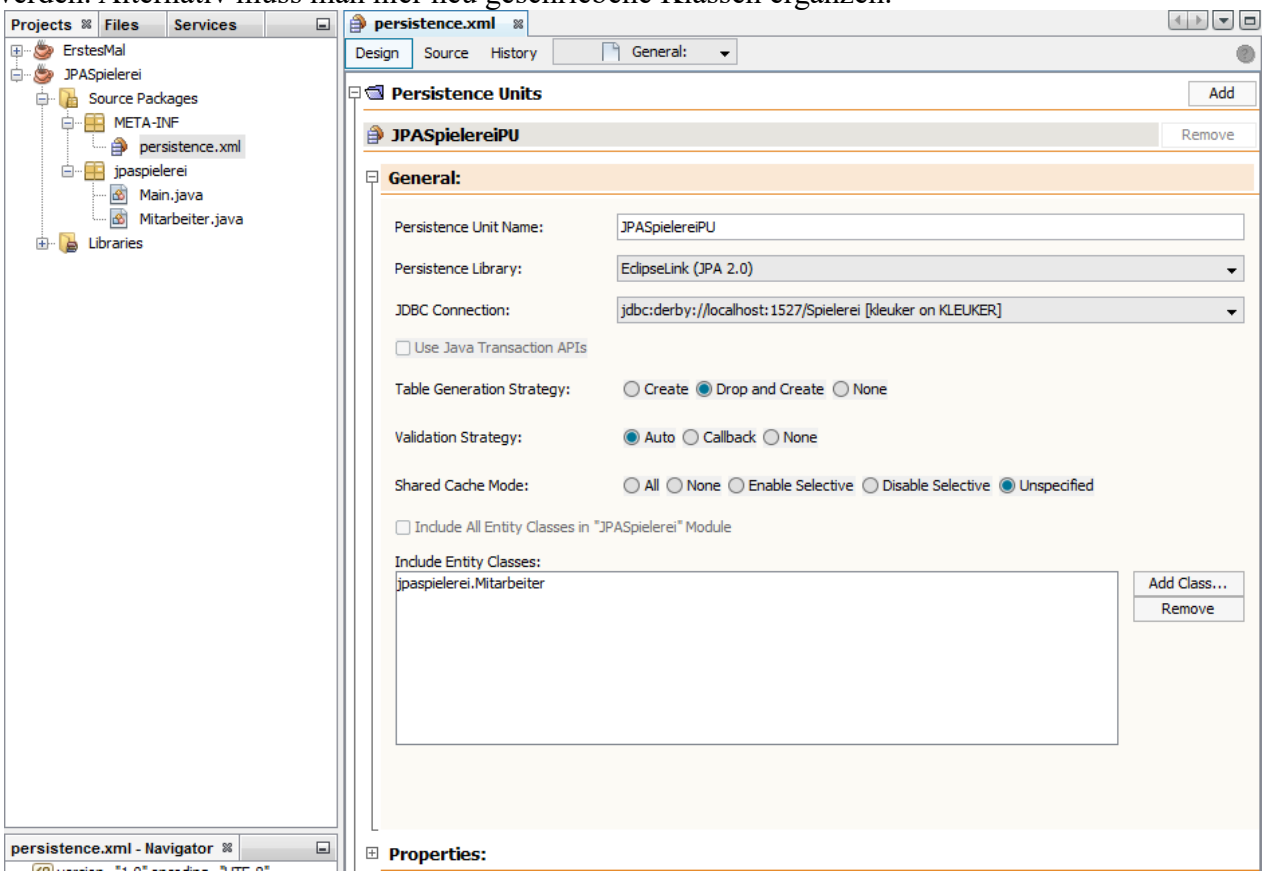
  @Override
  public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    if (!(object instanceof Mitarbeiter)) {
      return false;
```

```
    }
    Mitarbeiter other = (Mitarbeiter) object;
    if (this.id != other.id) {
      return false;
    }
    return true;
  }

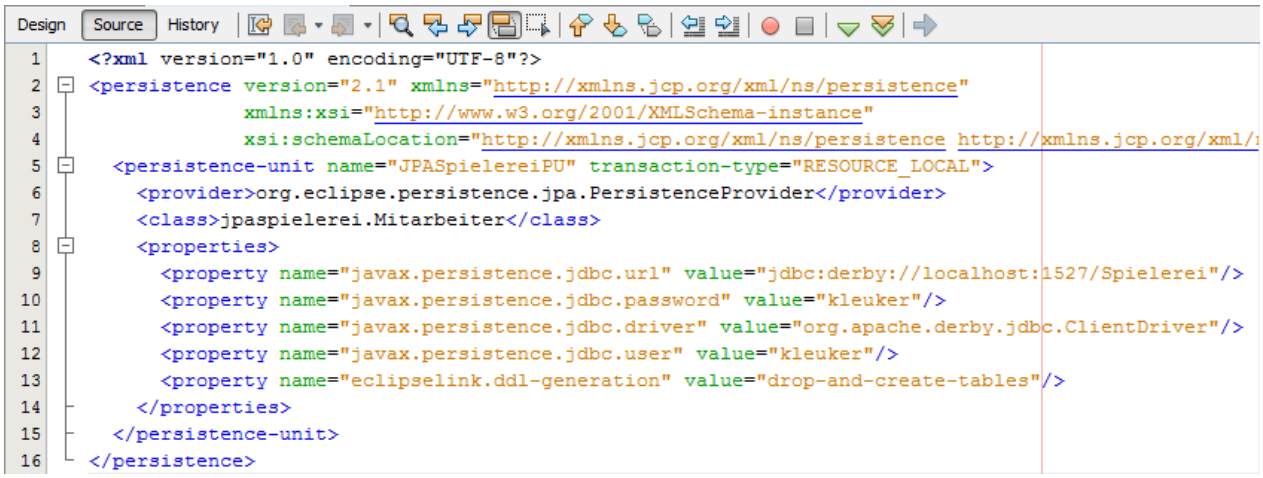
  @Override
  public String toString() {
    return "jpaspielerei.Mitarbeiter[ id=" + id + " ]";
  }

}
```

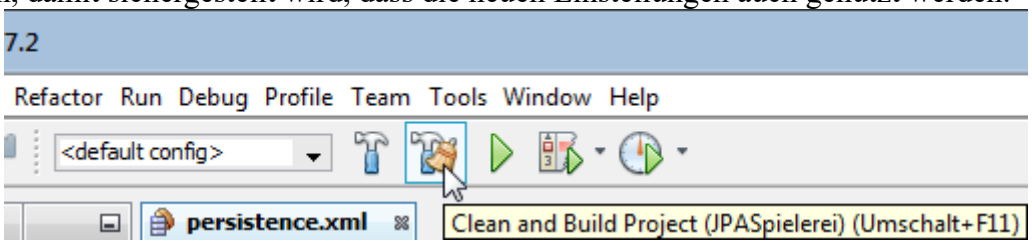
Nun muss die neue Klasse noch zur Verwaltung von JPA hinzugefügt werden, was in der Datei persistence.xml erfolgt. Durch einen Doppelklick auf die Datei wird ein Editor geöffnet, in dem weitere Einstellungen gemacht werden können. Diese Datei ist sonst auch direkt im Projekt-Ordner im Verzeichnis META-INF zugreifbar. Da die Entity Mitarbeiter über den Wizzard erzeugt wurde, muss diese Klasse nicht erst über „Add Class“ zu den von JPA verwalteten Klassen hinzugefügt werden. Alternativ muss man hier neu geschriebene Klassen ergänzen.



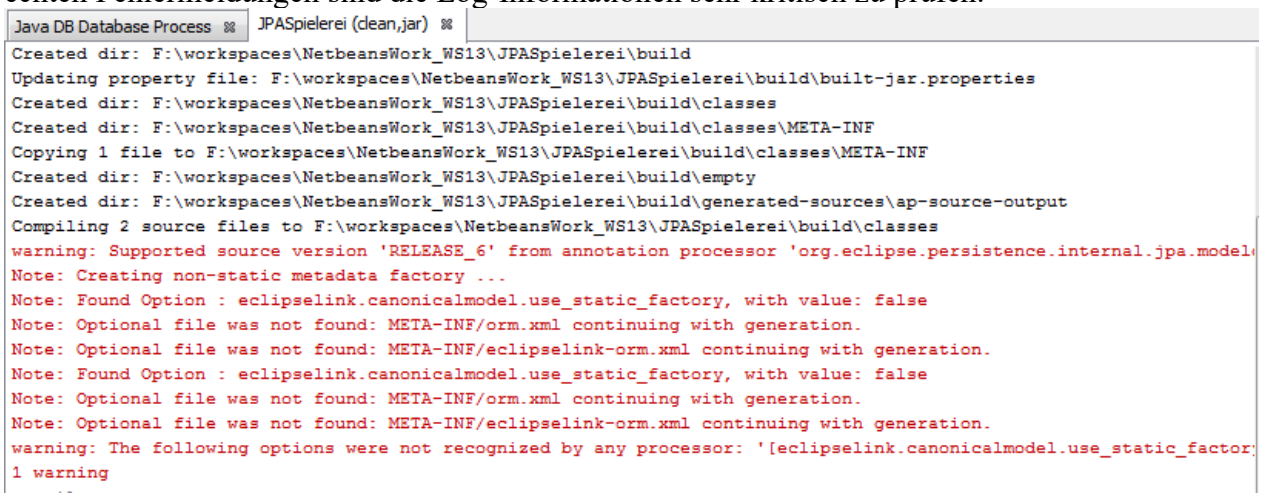
Möchte man nicht diesen Editor nutzen, kann man in der oberen Zeile in die Source-Ansicht wechseln und dort direkt die XML-Datei bearbeiten.



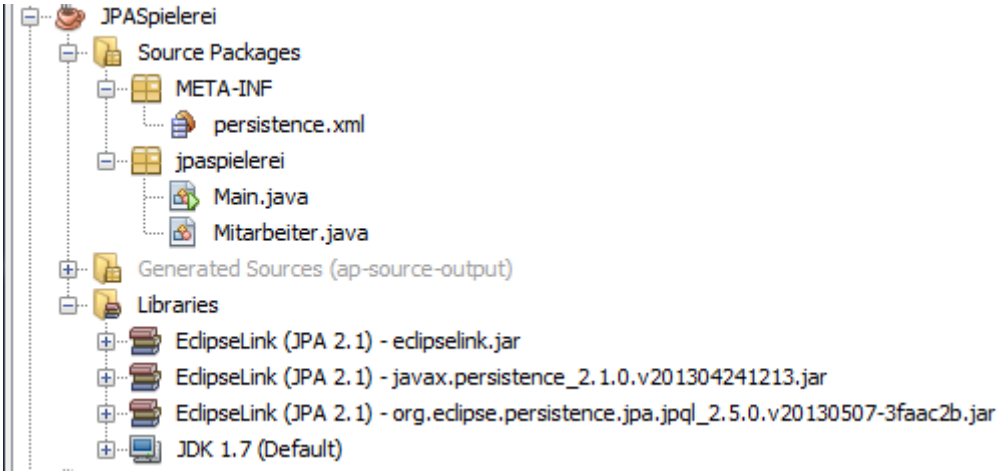
Sollte die Datei persistence.xml geändert werden, ist es generell sinnvoll, das gesamte Projekt neu zu übersetzen, damit sichergestellt wird, dass die neuen Einstellungen auch genutzt werden.



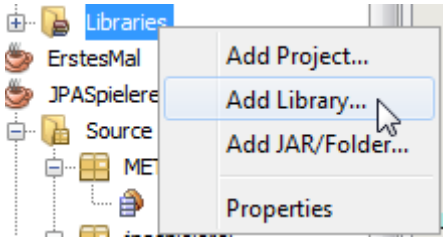
Etwaige rote Warnungen können gelesen, dann aber in diesem Fall auch ignoriert werden. Nur bei echten Fehlermeldungen sind die Log-Informationen sehr kritisch zu prüfen.



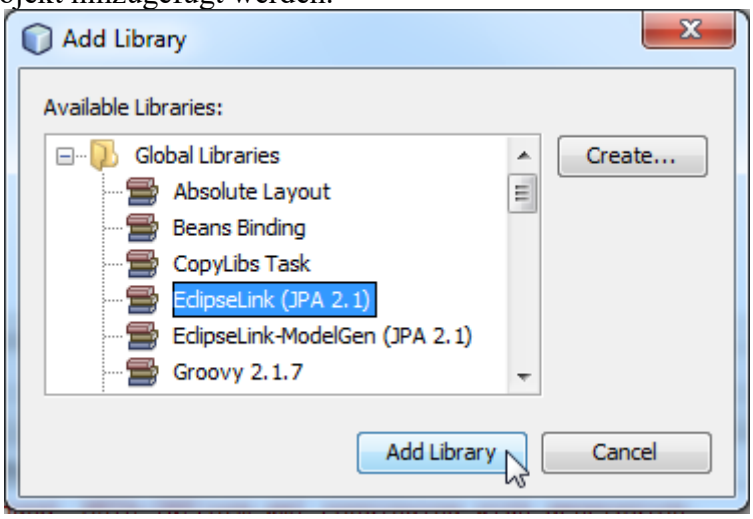
Schaut man sich den Projektordner an, so sind unter Libraries die für EclipseLink bzw. die eingesetzte JPA-Implementierung benötigten Bibliotheken ergänzt worden. Würde man die Persistenz-Datei nicht mit NetBeans erstellen, müssten man die benötigten Libraries selbst in das Projekt einbinden.



Sollen nicht individuelle Jar-Dateien genutzt werden, kann EclipseLink alternativ als Libraray von NetBeans genutzt werden. Dazu wird ein Rechtsklick auf „Libraries“ gemacht und „Add Library…“ gewählt.



Hier wird EclipseLink und dann „Add Library“ angeklickt, wodurch ebenfalls die vorher gezeigten Bibliotheken dem Projekt hinzugefügt werden.



Es fehlt allerdings noch eine JDBC-Realisierung für die genutzte Datenbank. ein dazu passender Weg wird im Abschnitt „4.6 Nutzung von Bibliotheken/Jar-Files (NetBeans 8)“ beschrieben. Das vollständige Projekt sieht dann wie folgt aus.



7.3 Nutzung eines EntityManagers

Möchte man die Entitys einer JPA-Implementierung nutzen, muss nur ein EntityManager-Objekt existieren, wie es das folgende Beispielprogramm zeigt. Das Beispiel hier zeigt dabei die Nutzung von JPA im Zusammenhang mit der Java SE (Standard Edition). Die Nutzung mit JEE wird später vorgestellt, dabei gibt es andere Wege zu einem EntityManager-Objekt zu kommen.
Das Programm erzeugt vier Mitarbeiter, persistiert diese in der Datenbank und liest dies dann wieder aus.

```
package jpaspielerei;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Main {

  private EntityManagerFactory emf =
    Persistence.createEntityManagerFactory("JPASpielereiPU");
  private EntityManager em = emf.createEntityManager();

  public void beispieldaten(){
    String namen[]={"Egon", "Erwin", "Ute", "Aise"};
    em.getTransaction().begin();
    for(String n:namen){
      Mitarbeiter m = new Mitarbeiter();
      m.setName(n);
      em.persist(m);
    }
    em.getTransaction().commit();
  }
```

```
    public void datenZeigen(){
      for (Mitarbeiter m: (List<Mitarbeiter>)em.
              createQuery("SELECT m FROM Mitarbeiter m").getResultList())
        System.out.println(m.getId()+": "+m.getName());
    }

    public void schliessen(){
      if(em!=null && em.isOpen())
        em.close();
      if(emf!=null && emf.isOpen())
        emf.close();
    }

    public static void main(String[] args) {
      Main m= new Main();
      m.beispieldaten();
      m.datenZeigen();
      m.schliessen();
    }
  }
```

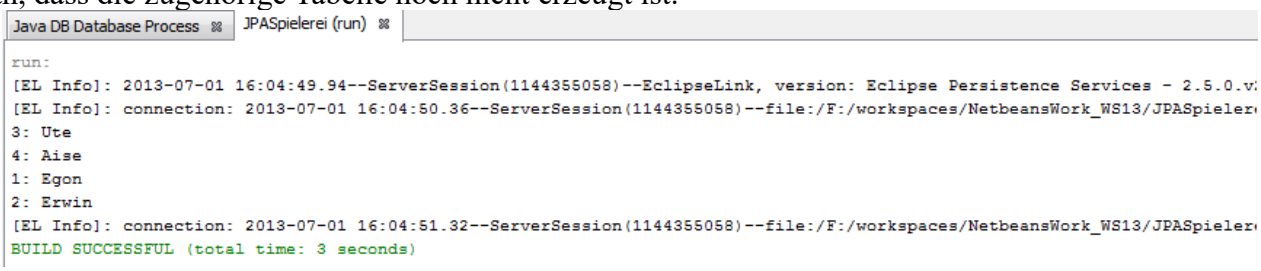
Erhält man beim Programmstart einen Fehler der folgenden Form, hat man wahrscheinlich den Namen der Persistence-Unit falsch geschrieben.

```
Exception in thread "main" javax.persistence.PersistenceException: No Persistence provider for EntityManager named JPASpielerei
        at javax.persistence.Persistence.createEntityManagerFactory(Persistence.java:55)
        at javax.persistence.Persistence.createEntityManagerFactory(Persistence.java:33)
```

Folgender Fehler zeigt, dass vergessen wurde den JDBC-Treiber mit in das Projekt einzubinden, wie es am Ende des vorherigen Abschnitts erwähnt wurde.

```
SCHWERWIEGEND: JDBC Driver class not found: org.apache.derby.jdbc.ClientDriver
java.lang.ClassNotFoundException: org.apache.derby.jdbc.ClientDriver
        at java.net.URLClassLoader$1.run(URLClassLoader.java:200)
        at java.security.AccessController.doPrivileged(Native Method)
```

Wird das Programm dann ausgeführt, erhält man folgendes Ergebnis. Die Fehlermeldung deutet nur an, dass die zugehörige Tabelle noch nicht erzeugt ist.



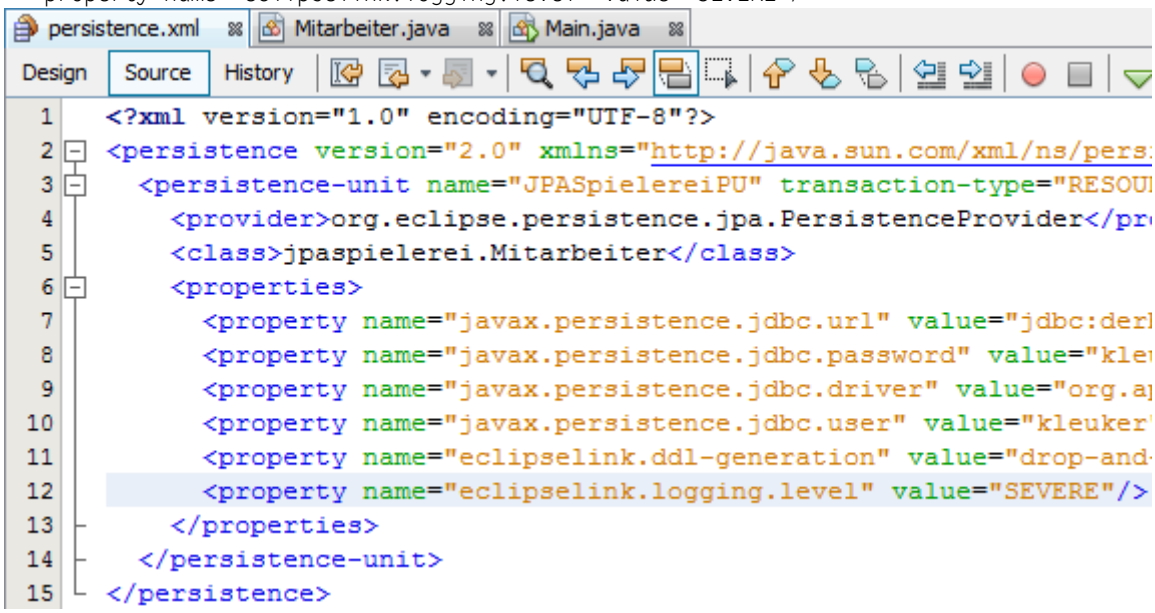
Der genaue Text der vorherigen Abbildung lautet:

```
run:
[EL Info]: 2013-07-01 16:04:49.94--ServerSession(1144355058)--EclipseLink, version:
Eclipse Persistence Services - 2.5.0.v20130507-3faac2b
```

```
[EL Info]: connection: 2013-07-01 16:04:50.36--ServerSession(1144355058)--
file:/F:/workspaces/NetBeansWork_WS13/JPASpielerei/build/classes/_JPASpielereiPU login
successful
3: Ute
4: Aise
1: Egon
2: Erwin
[EL Info]: connection: 2013-07-01 16:04:51.32--ServerSession(1144355058)--
file:/F:/workspaces/NetBeansWork_WS13/JPASpielerei/build/classes/_JPASpielereiPU logout
successful
BUILD SUCCESSFUL (total time: 3 seconds)
```

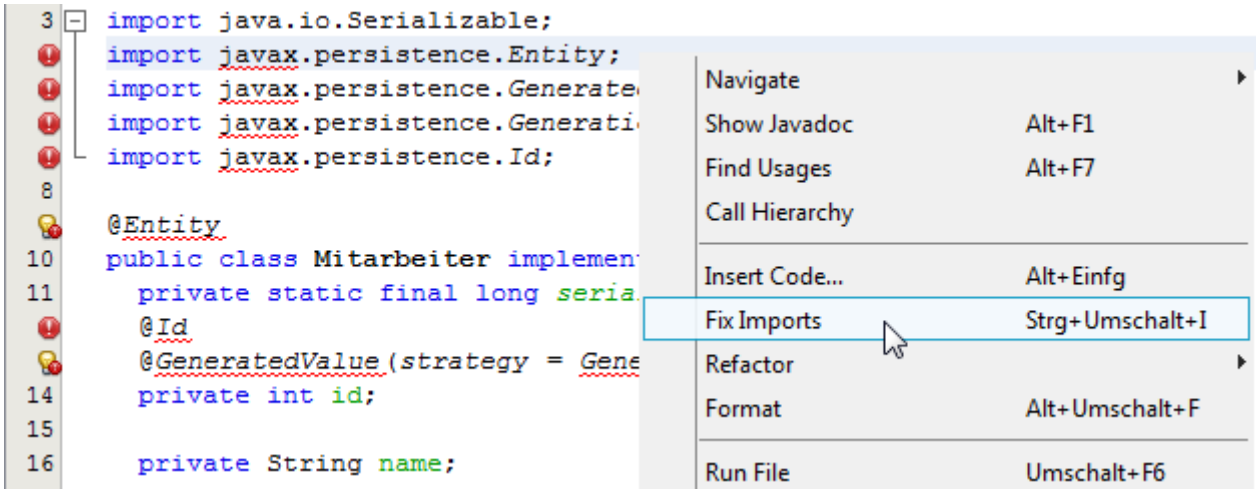
Falls man die Logging-Informationen von EclipseLink bis auf echte Fehler abschalten will, muss man in der persistence.xml-Datei nur die folgende Zeile ergänzen.

```
<property name="eclipselink.logging.level" value="SEVERE"/>
```

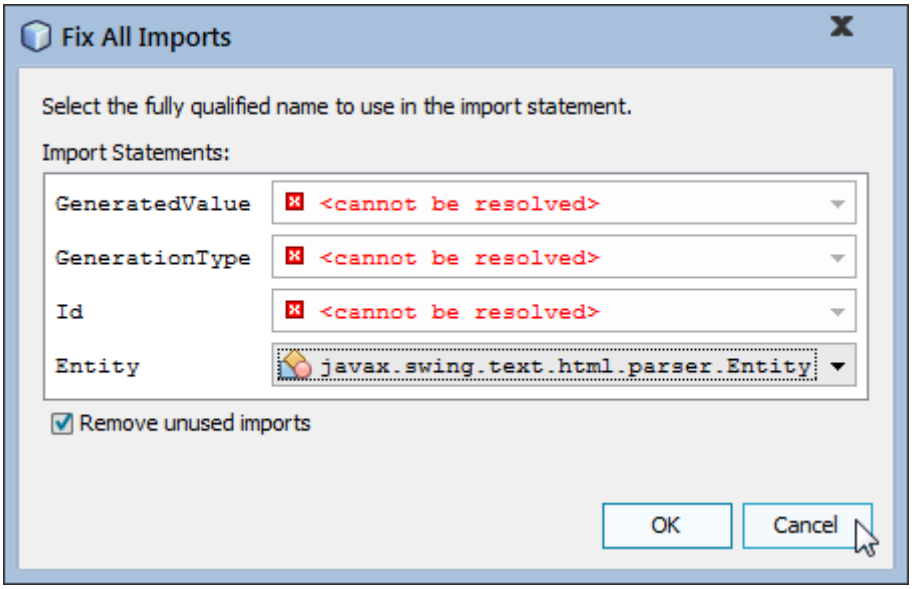


7.4 Umwandlung von Klassen in JPA-Entity-Klassen

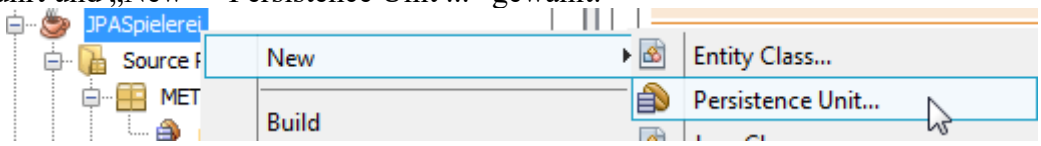
Generell können Entity-Dateien als normale *.java-Dateien behandelt werden. Beim Einspielen dieser Dateien in ein neues Projekt ohne bisherige JPA-Nutzung steht man vor dem Problem, dass man die Abhängigkeiten auflösen muss.



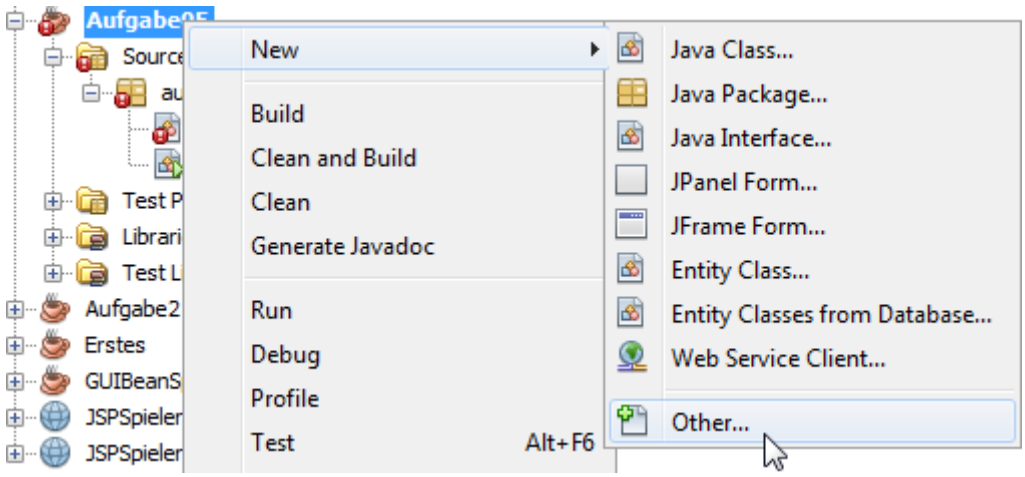
Versucht man dies über NetBeans mit „Fix Imports“ so erhält man, wie in der folgenden Abbildung gezeigt, leider keine Lösung angeboten.



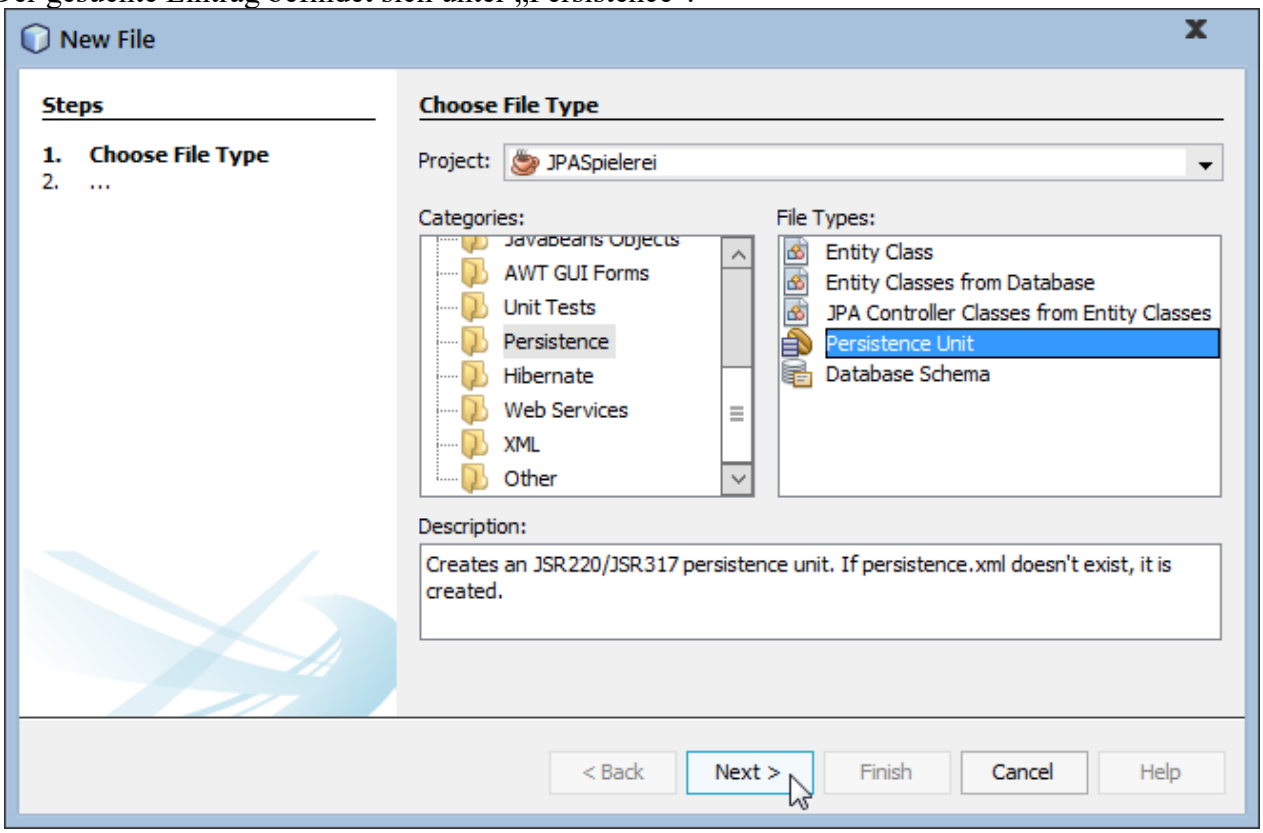
Es besteht die Notwendigkeit, die Einbindung einer JPA-Realisierung nachträglich hinzuzufügen. Dieser Ansatz wird direkt von NetBeans unterstützt. Dazu wird ein Rechtsklick auf dem Projekt durchgeführt und „New -> Persistence Unit ...“ gewählt.



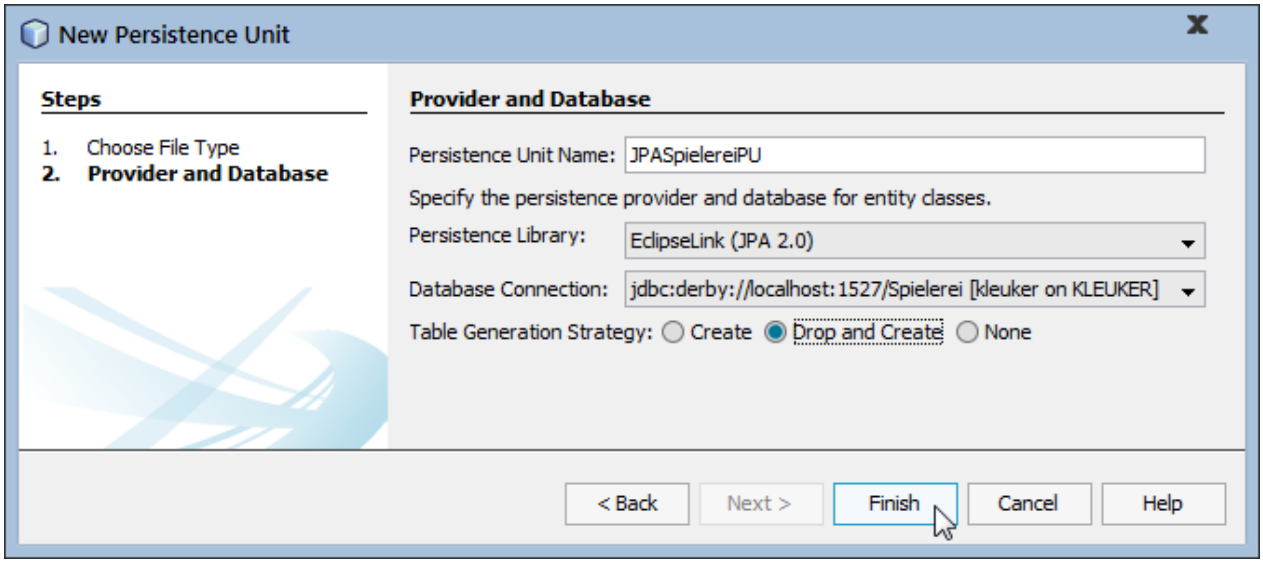
Falls Persistence Unit nicht sofort angeboten wird, muss der Eintrag unter „Other…“ gesucht werden.



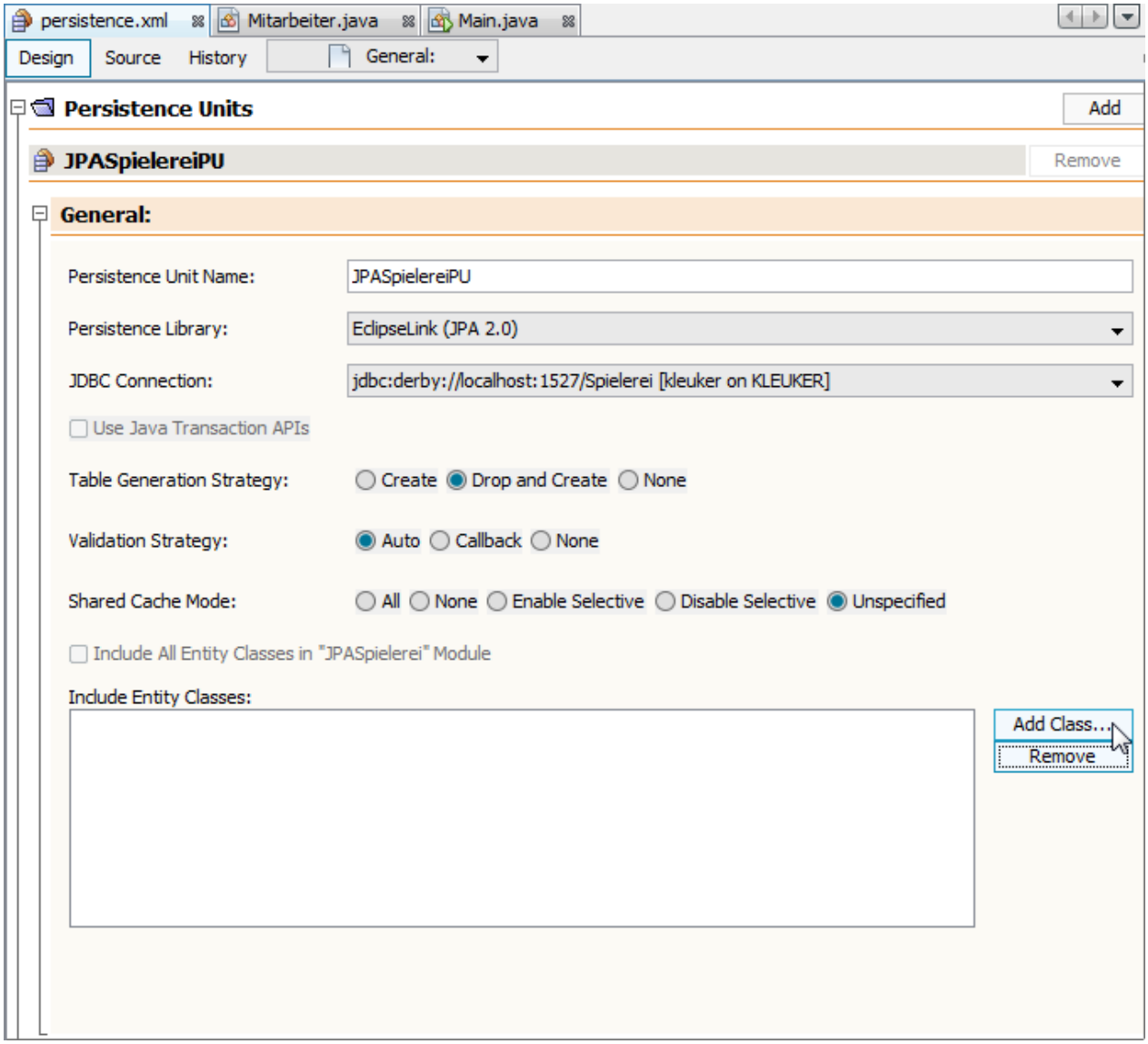
Der gesuchte Eintrag befindet sich unter „Persistence“.



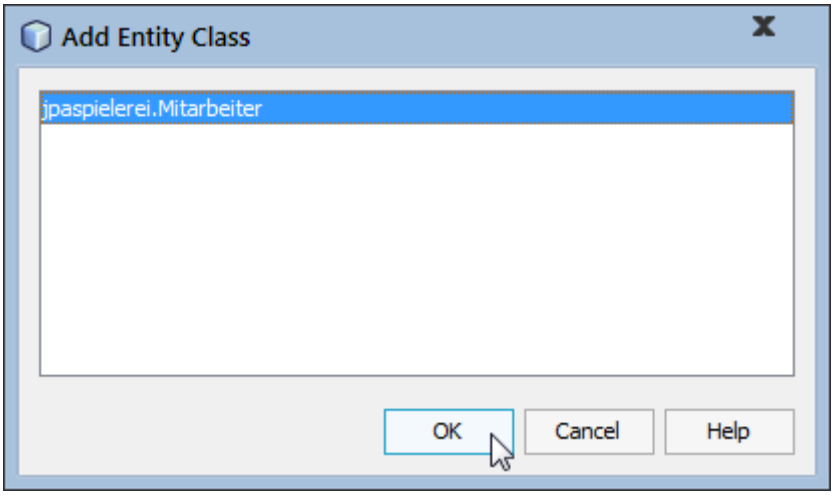
Der Eintrag „Persistence Unit“ führt zu einem Fenster, dessen Behandlung bereits unter „Neue Entities anlegen“ beschrieben wurde (dort „Create Persistence Unit“).



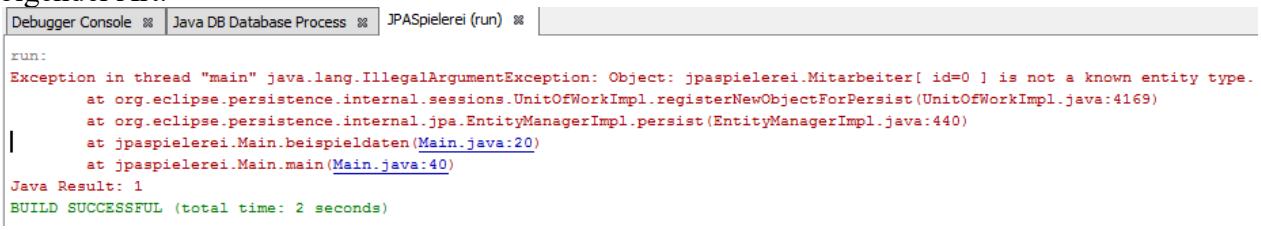
Im Editor zur Bearbeitung der persistence.xml muss dann nur die kopierte Klasse über „Add Class“ in die Persistence Unit aufgenommen werden.



Das Projekt wird nach nutzbaren Klassen durchsucht, so dass man die eingespielte Klasse jetzt einbinden und wie andere Entity-Klassen behandeln kann.



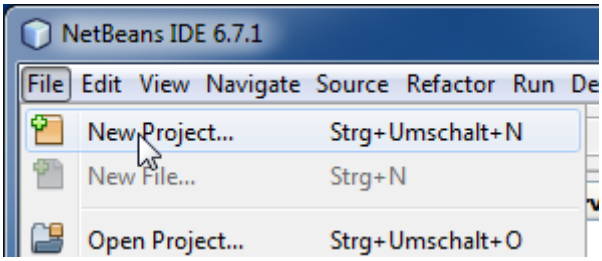
Sollte man vergessen, eine Klasse in persistence.xml einzutragen, erhält man eine Fehlermeldung folgender Art.



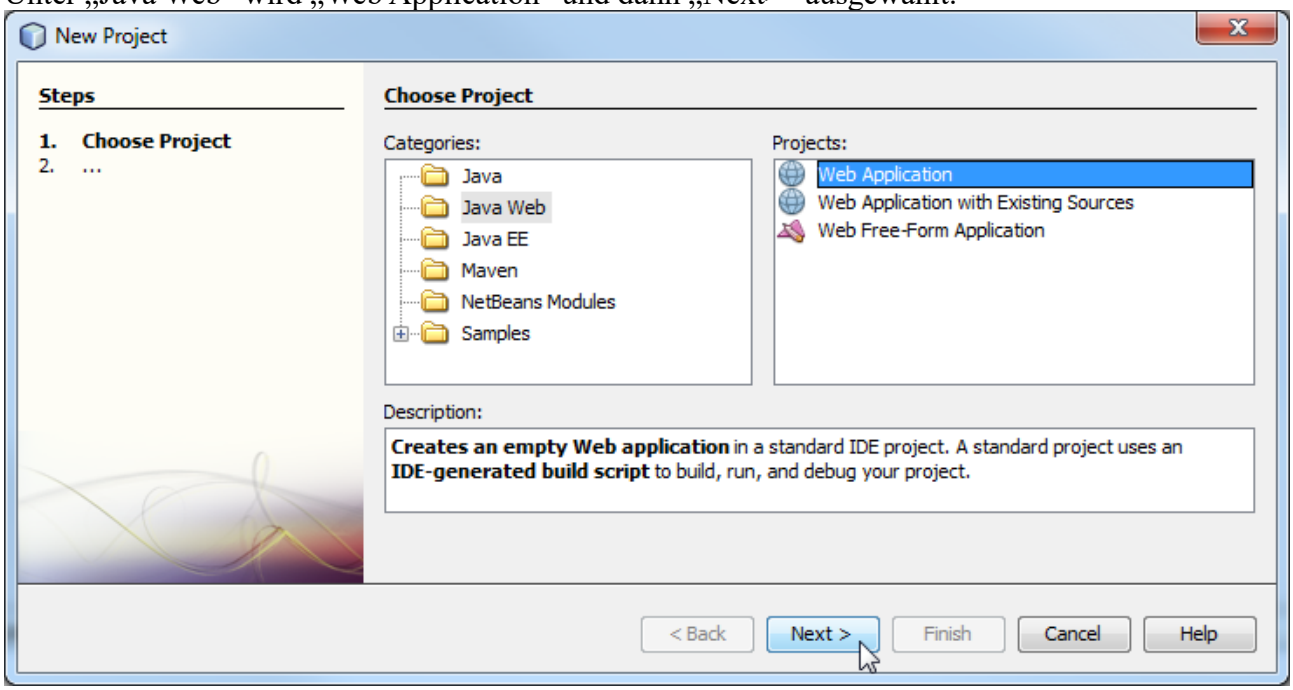
8 JSP (NetBeans 6.7)

8.1 Erstellung eines JSP-Projekts

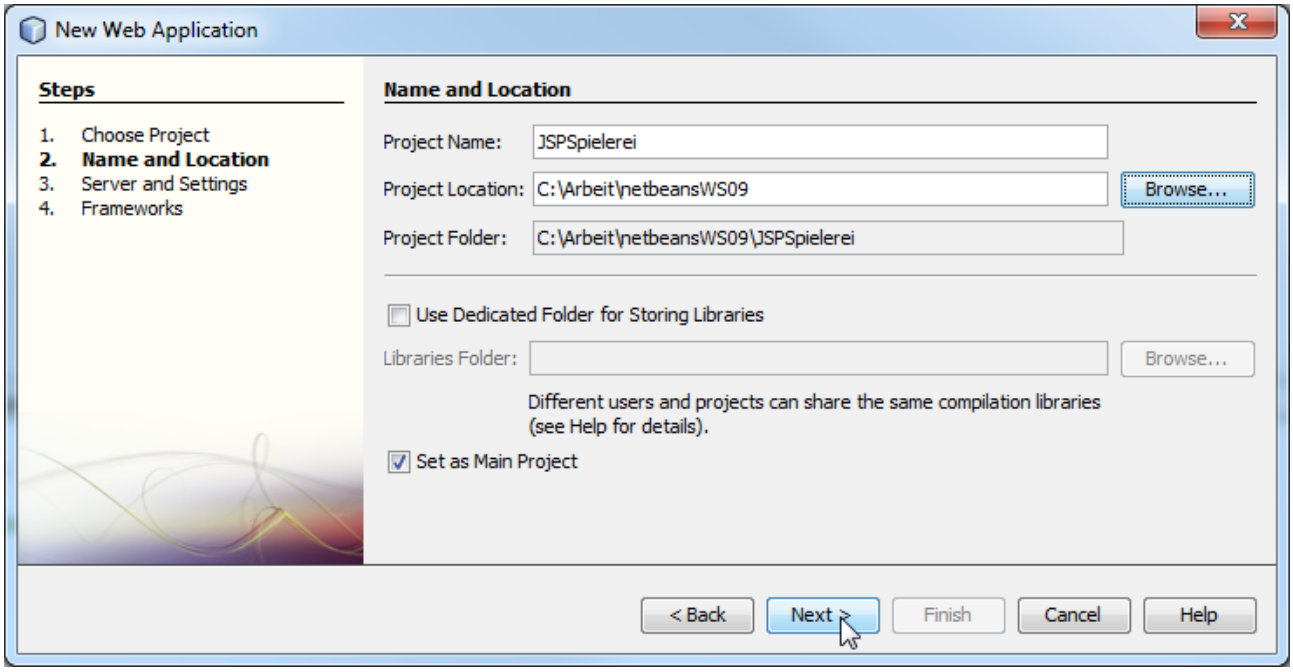
Java ServerPages können mit NetBeans sehr einfach erstellt werden, da automatisch ein Projekt mit der richtigen Ordnerstruktur erstellt wird. Dies ist zum Üben sehr hilfreich, trotzdem sollte man sich informieren, was bei dem gesamten Prozess passiert.
Man legt ein neues Projekt an.



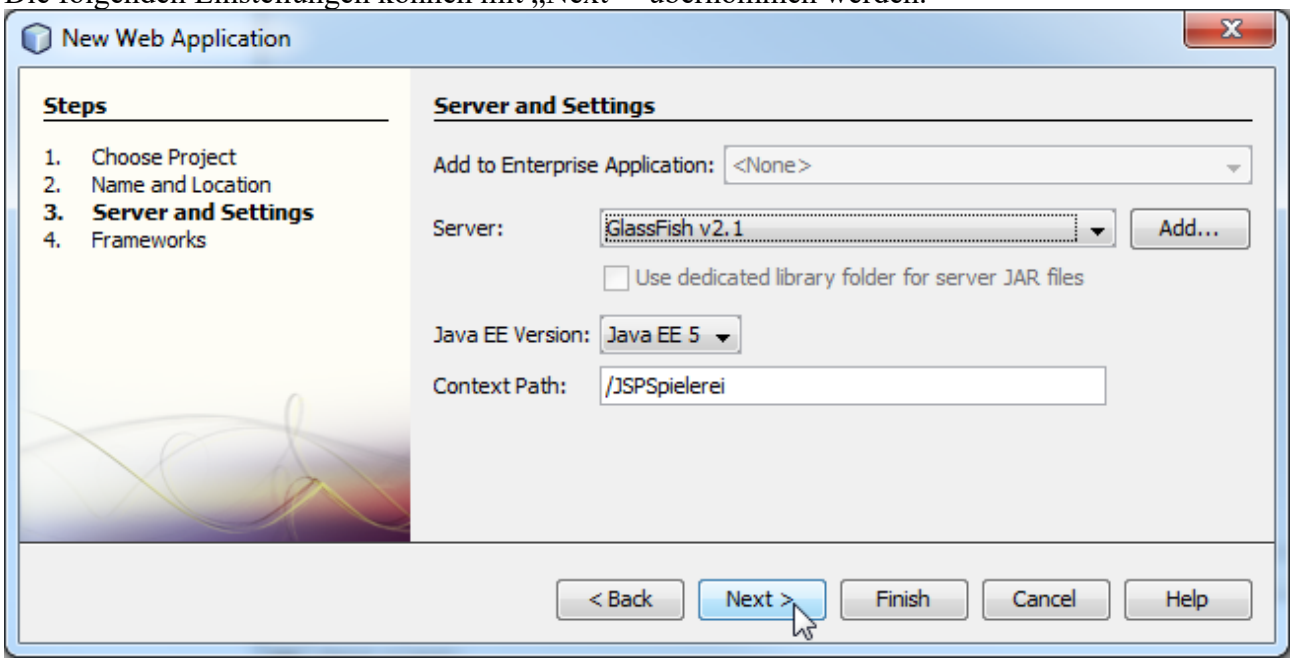
Unter „Java Web“ wird „Web Application“ und dann „Next>“ ausgewählt.



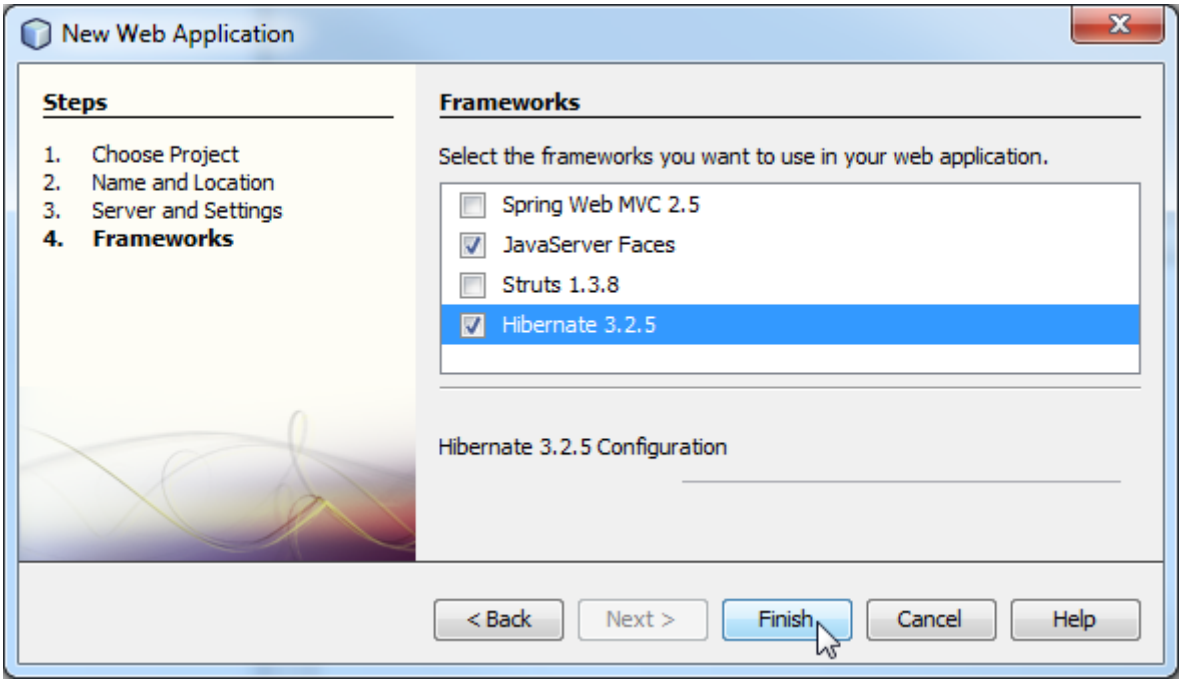
Das Projekt erhält einen Namen und es wird „Next>“ gedrückt.



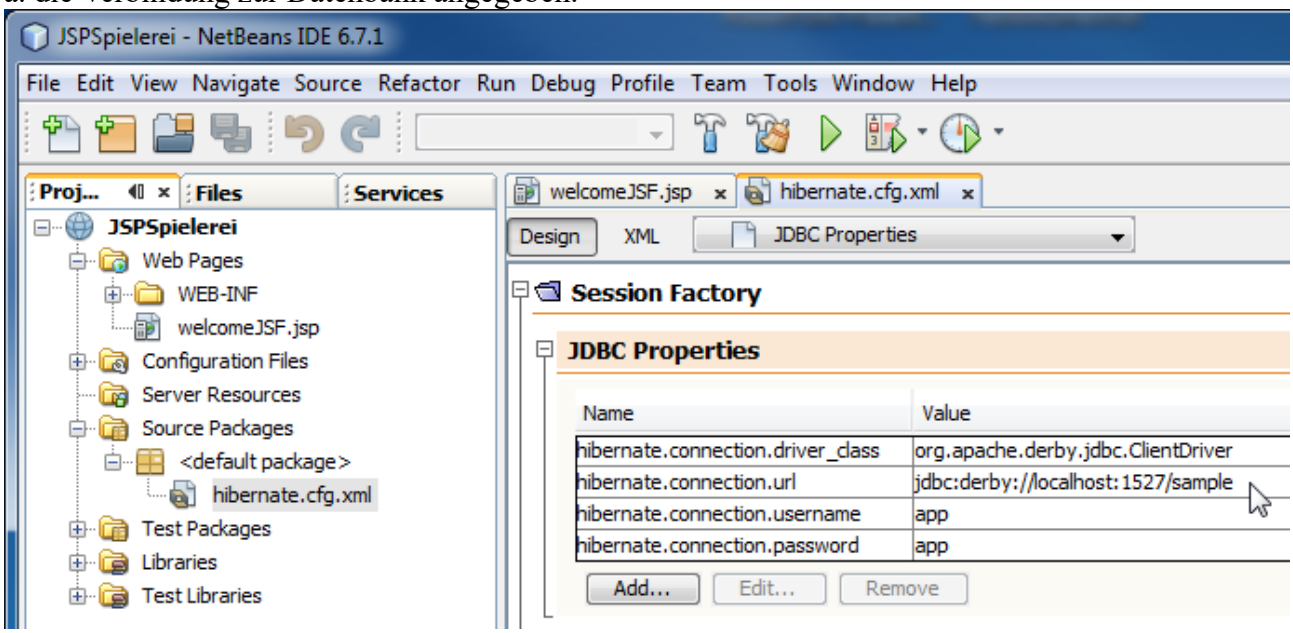
Die folgenden Einstellungen können mit „Next>“ übernommen werden.



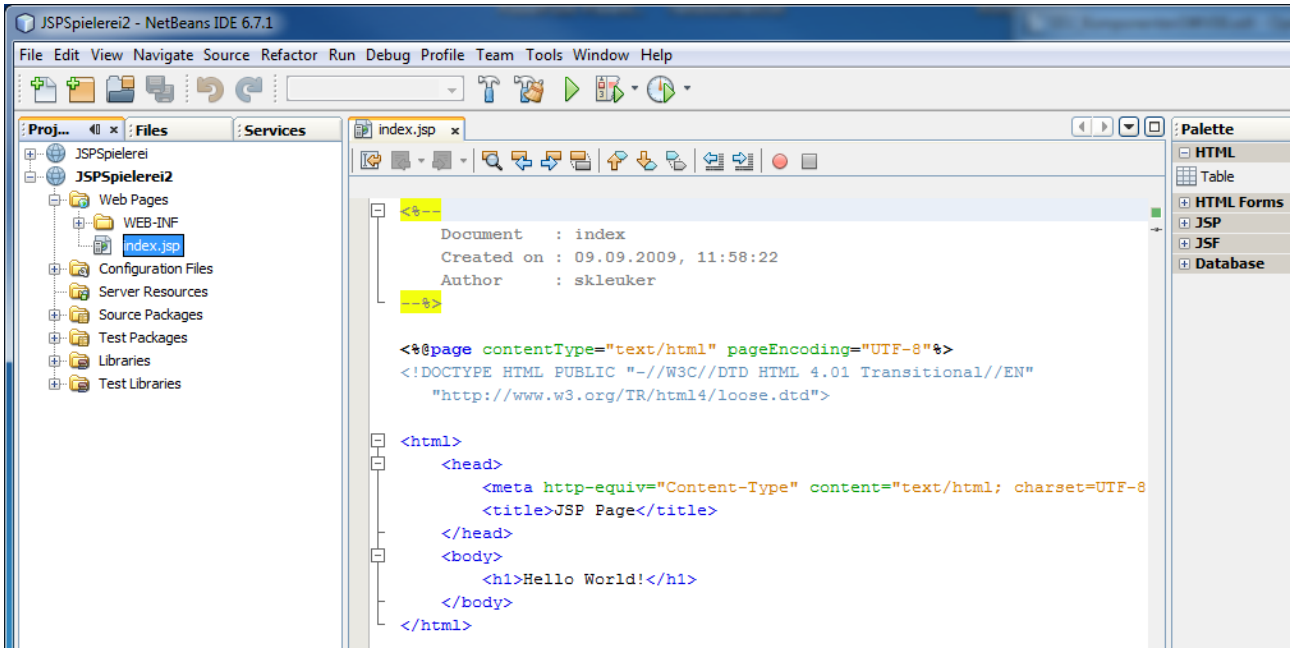
Die Haken im folgenden Bild hängen vom Projekt ab. Die gesetzten Haken wird man in der Lehrveranstaltung häufiger nutzen. Möchte man nur JSP-Seiten erstellen, werden hier keine Haken, zumindest nicht bei JavaServerFaces gesetzt, da sonst automatisch eine JSF-Seite zum Start generiert wird. Die Projekterstellung wird mit „Finish“ abgeschlossen.



Das folgende Bild zeigt als Ausblick auf die JSF-Entwicklung das angelegte JSF-Projekt. Man erkennt sofort, dass die Hibernate- Einstellungen in der Datei hibernate.cfg.xml stehen. Hier wird u. a. die Verbindung zur Datenbank angegeben.



Hat man kein JSF ausgewählt, sieht das leere Projekt, hier „JSPSpielerei2“ wie folgt aus.



8.2 Deployment und Ausführung eines einfachen Projekts

Hier wird von einem einfachen JSP-Projekt ausgegangen. Es soll folgende Seite index.jsp existieren.

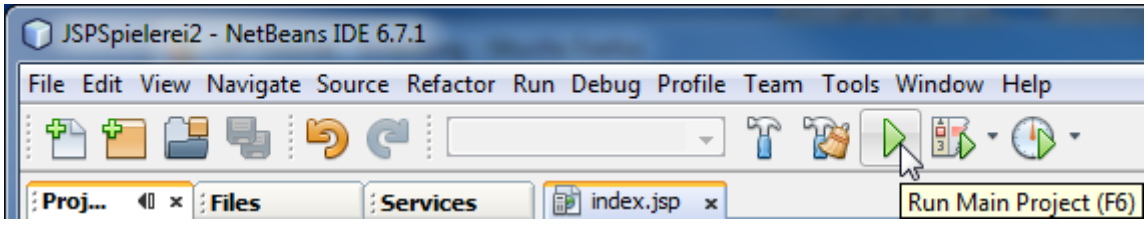
index.jsp

```
<%--
    Document   : index
    Created on : 09.09.2009, 11:58:22
    Author     : skleuker
--%>

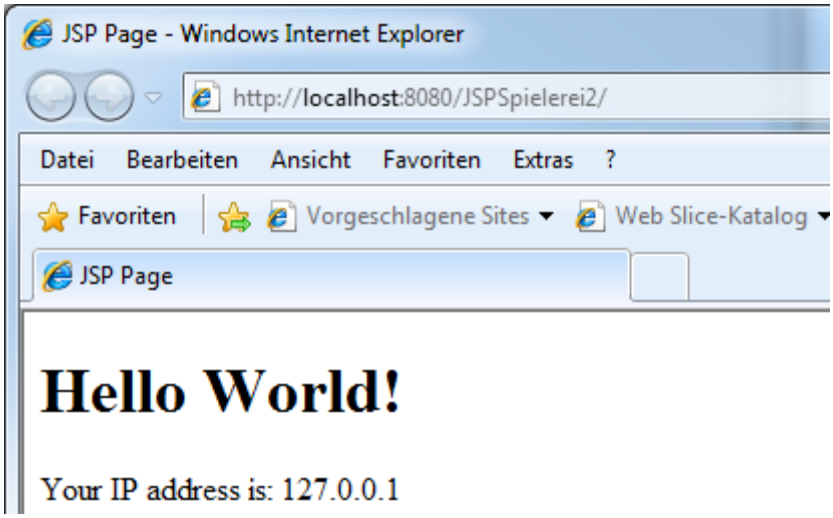
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        Your IP address is: <%= request.getRemoteAddr() %><br>
    </body>
</html>
```

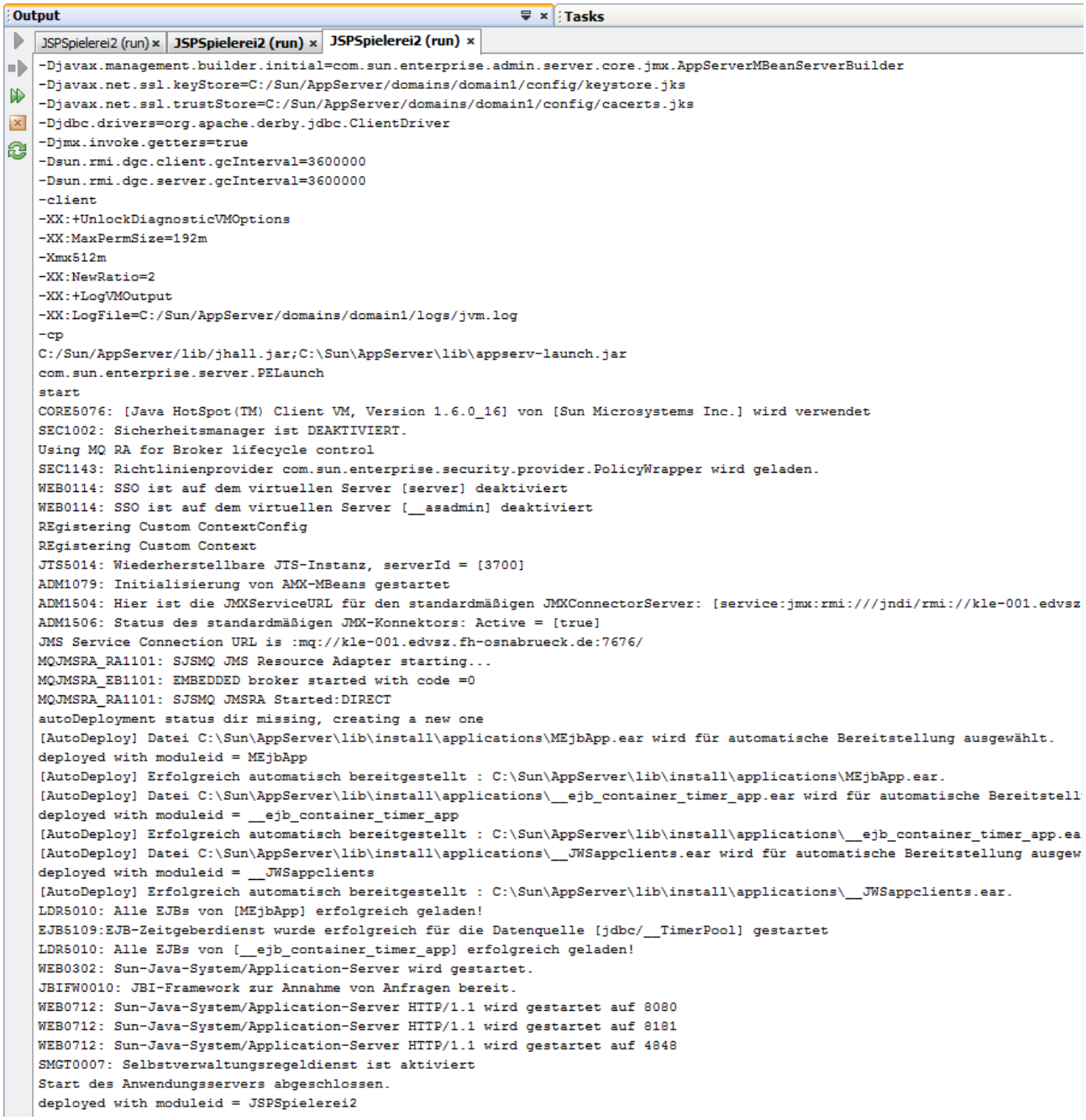
Nun wird das Projekt über das grüne Dreieck gestartet. Hat man vorher noch keine Schritte unternommen, passieren jetzt einige Schritte. Es werden die Seiten übersetzt, der GlassFish-Server gestartet, die Seiten übertragen (deployt) und zur Ausgabe der Standard-Internetseiten-Betrachter aufgerufen.



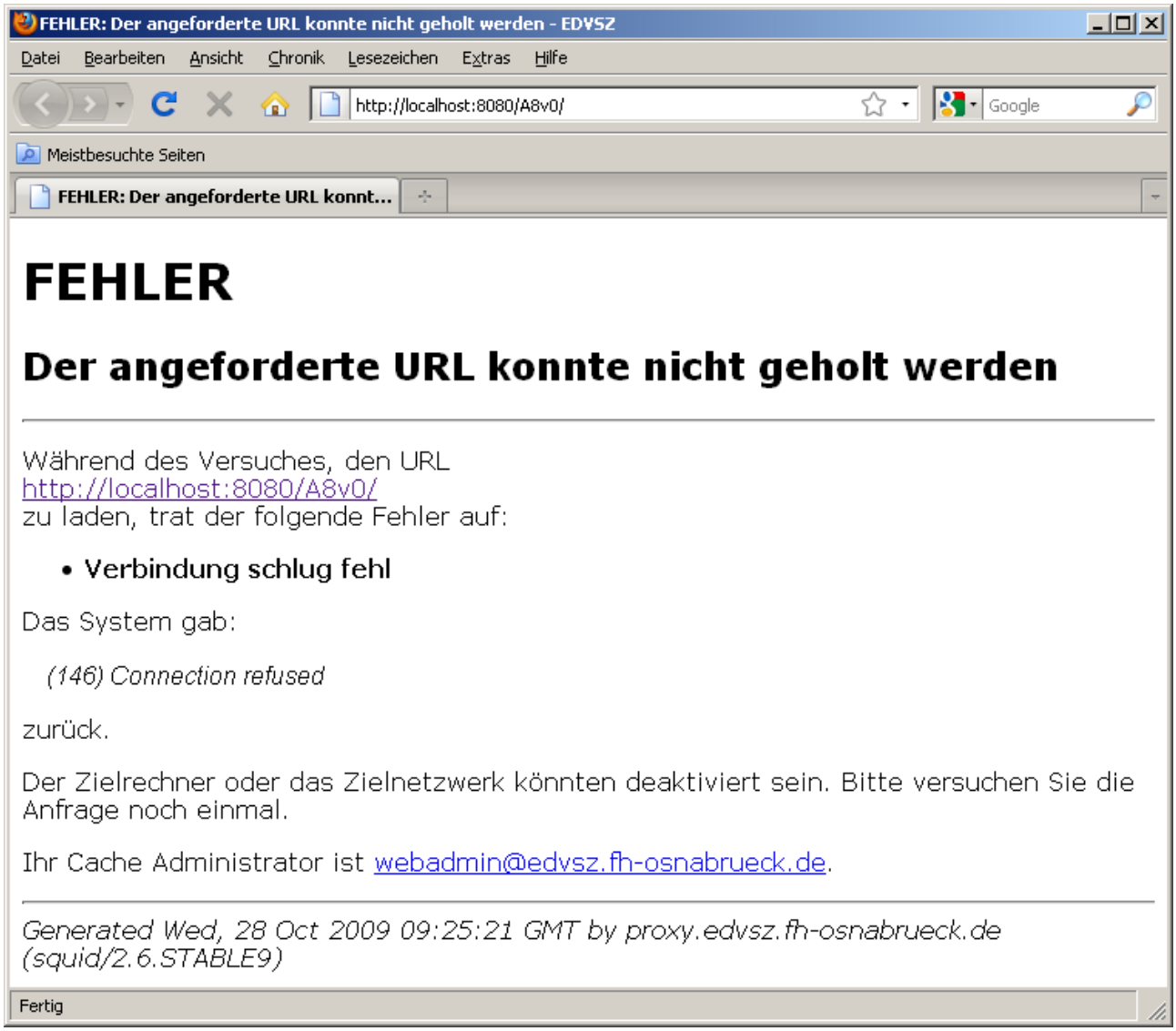
Man erhält z. B. folgende Ausgabe.



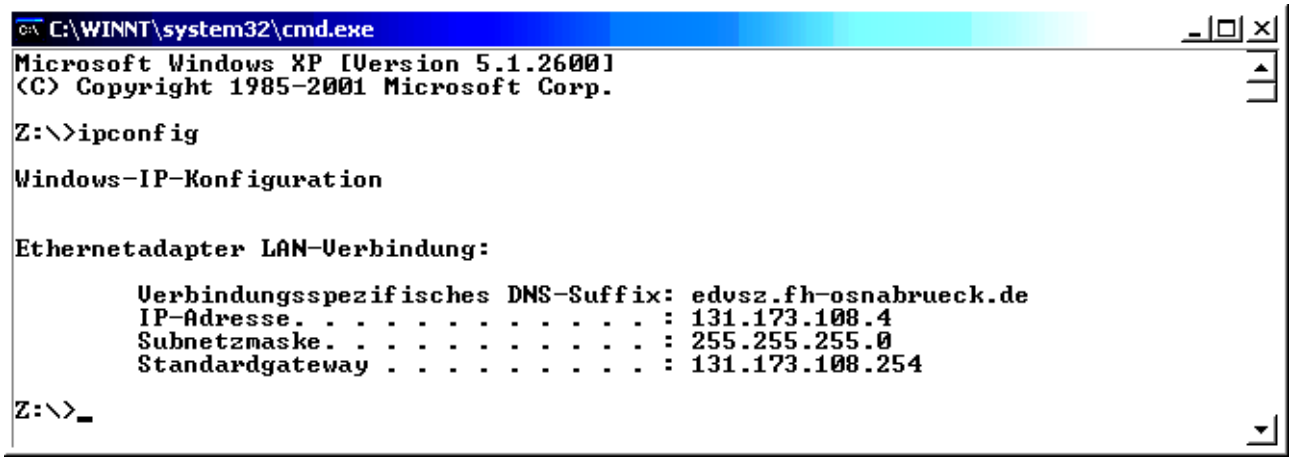
Unten bei den Output-Fenstern erhält man Informationen über den gestarteten Server, die wie folgt aussehen können.



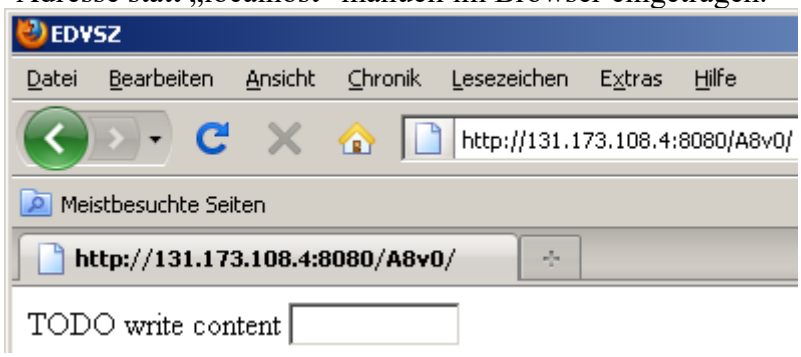
In seltenen Fällen kann es passieren, dass der Zugriff auf localhost nicht funktioniert, wie es in der folgenden Abbildung zu sehen ist.



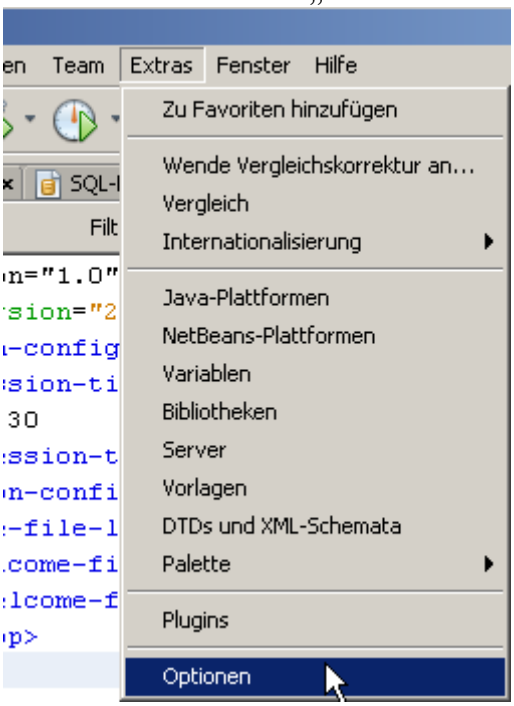
In diesem Fall muss man die Angabe “localhost“ durch die IP-Adresse ersetzt werden, die man unter Windows z. B. durch den Befehl ipconfig bestimmen kann.



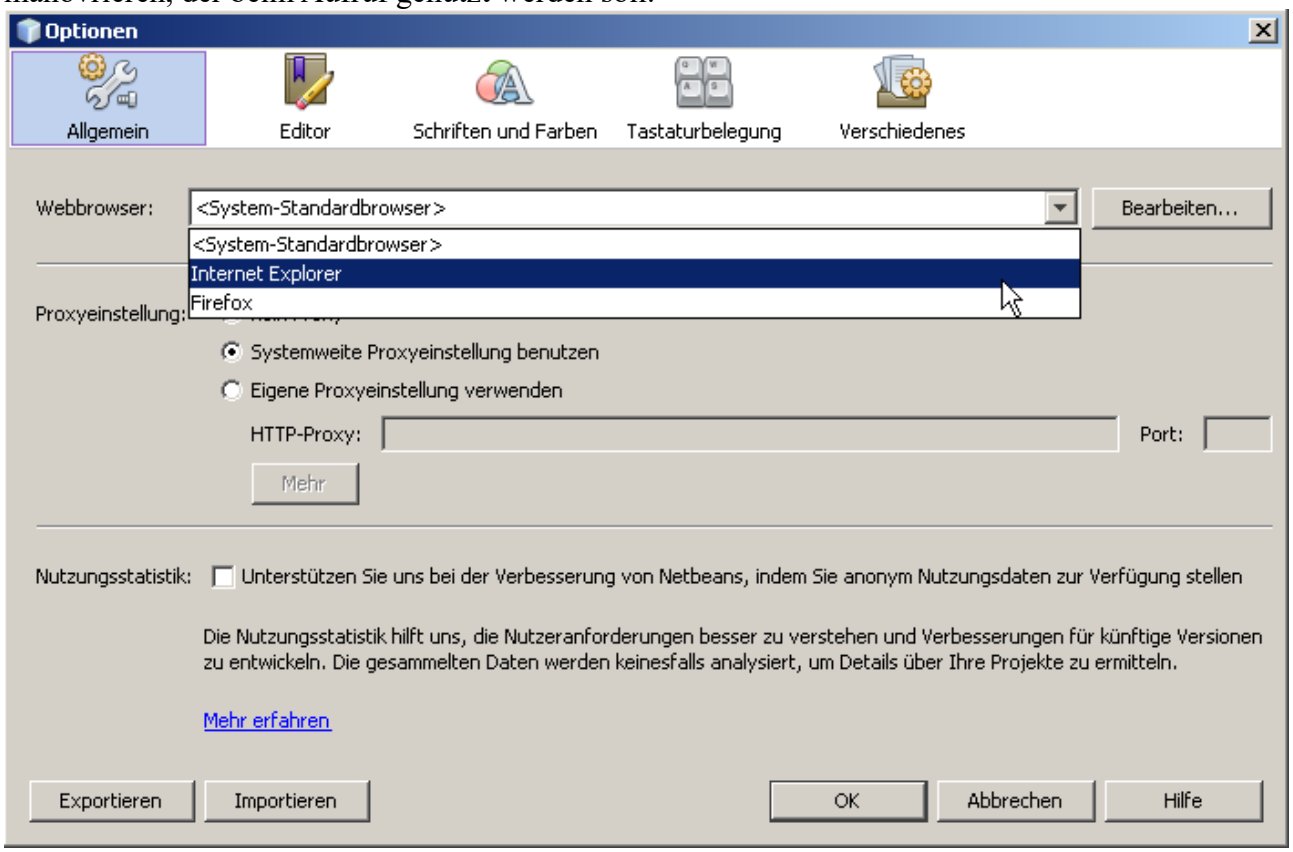
Danach wird die IP-Adresse statt „localhost“ manuell im Browser eingetragen.



NetBeans ermöglicht die Auswahl einer der installierten Browser als Browser, der beim Aufruf der Web-Seite genutzt wird. In NetBeans wird dazu unter „“ der untere Punkt „Options“ gewählt.

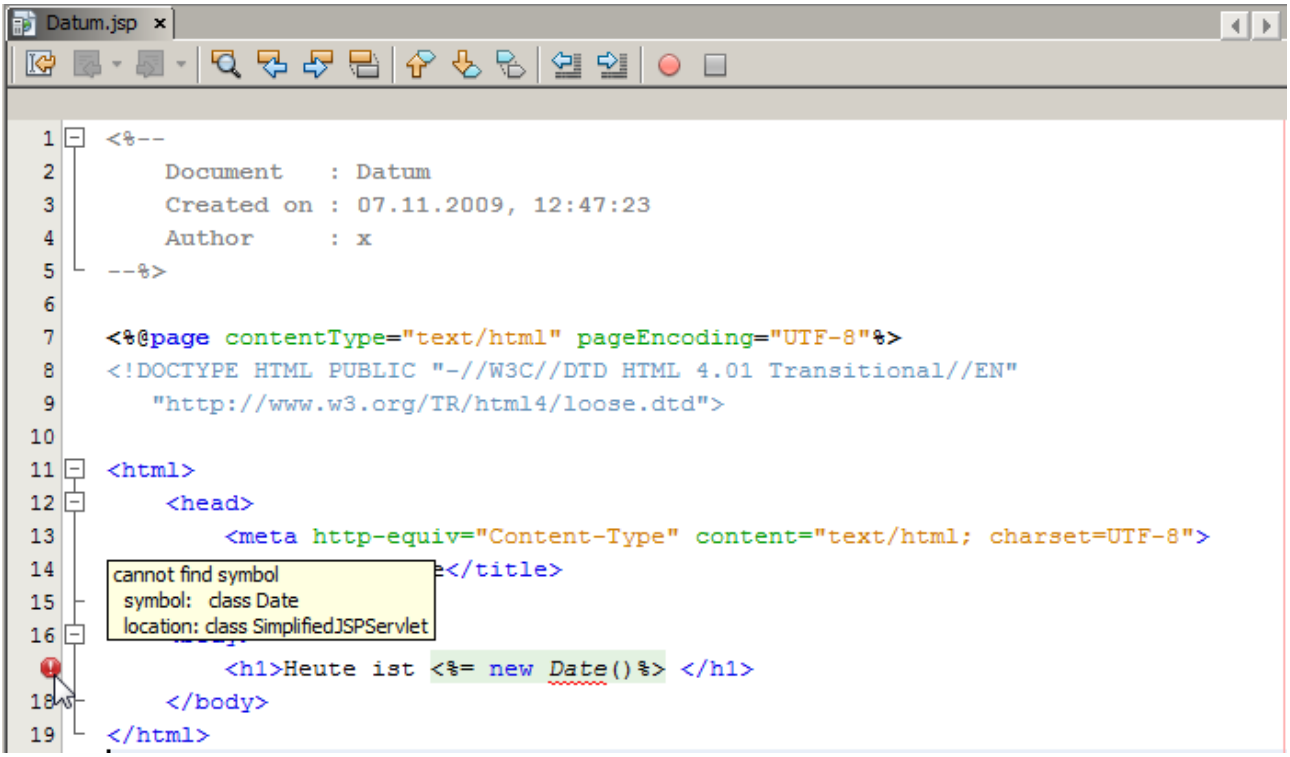


Im Auswahlpunkt „Allgemein“ oben kann dann im Auswahlmenü für Webbrowser einer der installierten Browser ausgewählt werden. Alternativ kann man unter „Bearbeiten“ zu einem Browser manövrieren, der beim Aufruf genutzt werden soll.



8.3 Fehler in JSPs korrigieren

Da JSPs in Servlets (nächstes Kapitel) und damit Java-Klassen übersetzt werden, kann das Auffinden von Programmierfehlern aufwändig werden. Syntaxfehler werden dagegen bereits von NetBeans recht zuverlässig erkannt, wobei die Seite eventuell erst gespeichert werden muss. Das folgende Beispiel zeigt einen vergessenen Import in der page-Direktive.

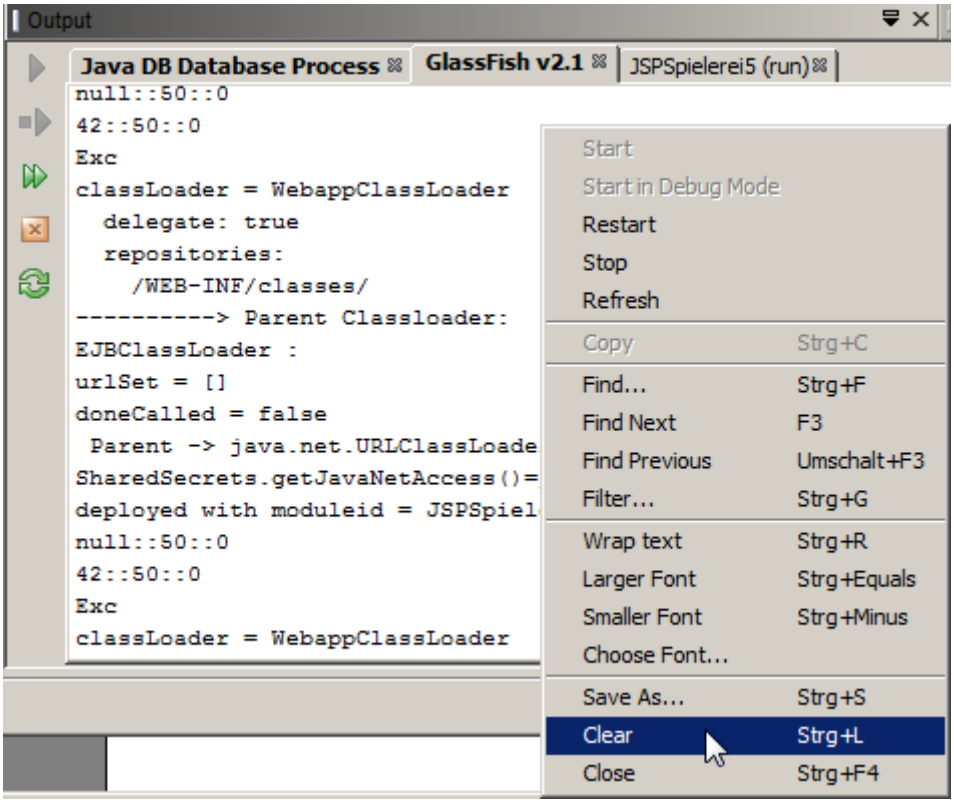


Der obige Fehler ist in der folgenden Abbildung korrigiert, allerdings wird hier vergessen, dass der Parameter evtl. nicht existiert und das Methodenergebnis zu einer Null-Pointer-Nutzung führen könnte.

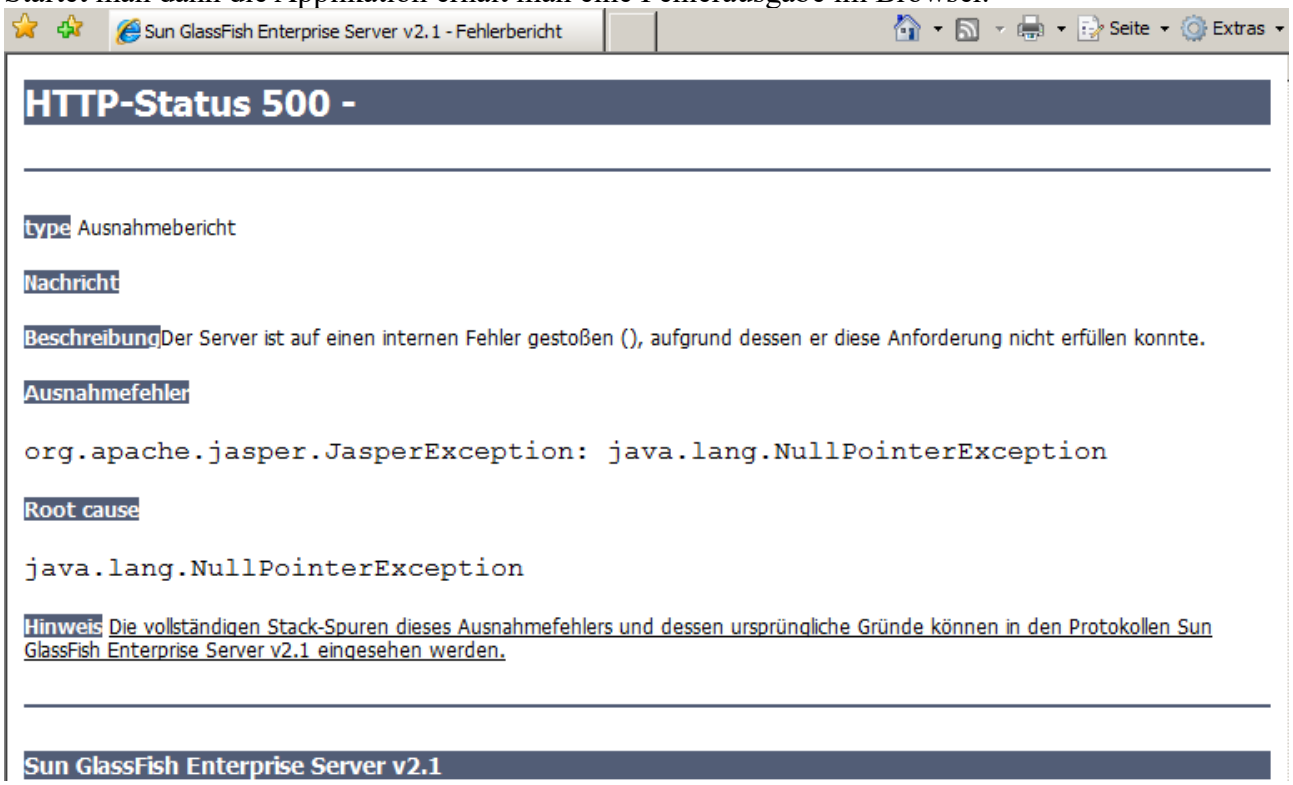
Datum.jsp

```
1  <%--
2      Document   : Datum
3      Created on : 07.11.2009, 12:47:23
4      Author     : x
5  --%>
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"
8          import="java.util.Date"%>
9  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
10   "http://www.w3.org/TR/html4/loose.dtd">
11
12 <html>
13   <head>
14     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15     <title>JSP Page</title>
16   </head>
17   <body>
18     <h1>Heute ist <%= new Date()%> </h1>
19     <% String s=request.getParameter("nichtDa");%>
20     <h2> Anzahl Zeichen: <%=s.length()%> </h2>
21   </body>
22 </html>
```

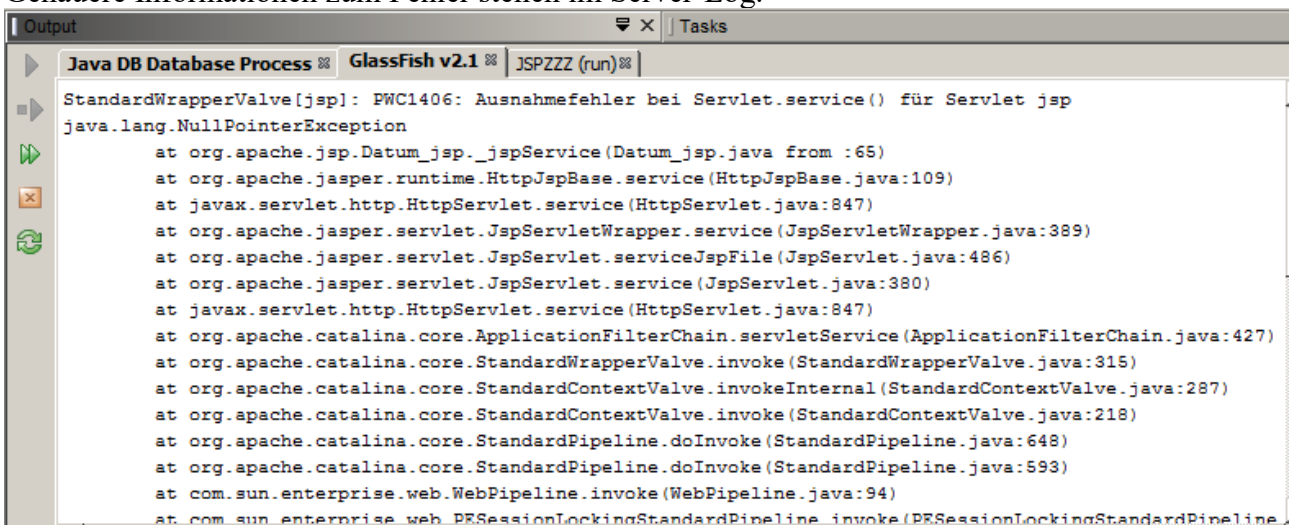
Generell ist es eine gute Idee, die bisherigen Ausgaben des Servers zu löschen. Dazu geht man unten in das GlassFish-Ausgabefenster, macht einen Rechtsklick und wählt „Clear“.



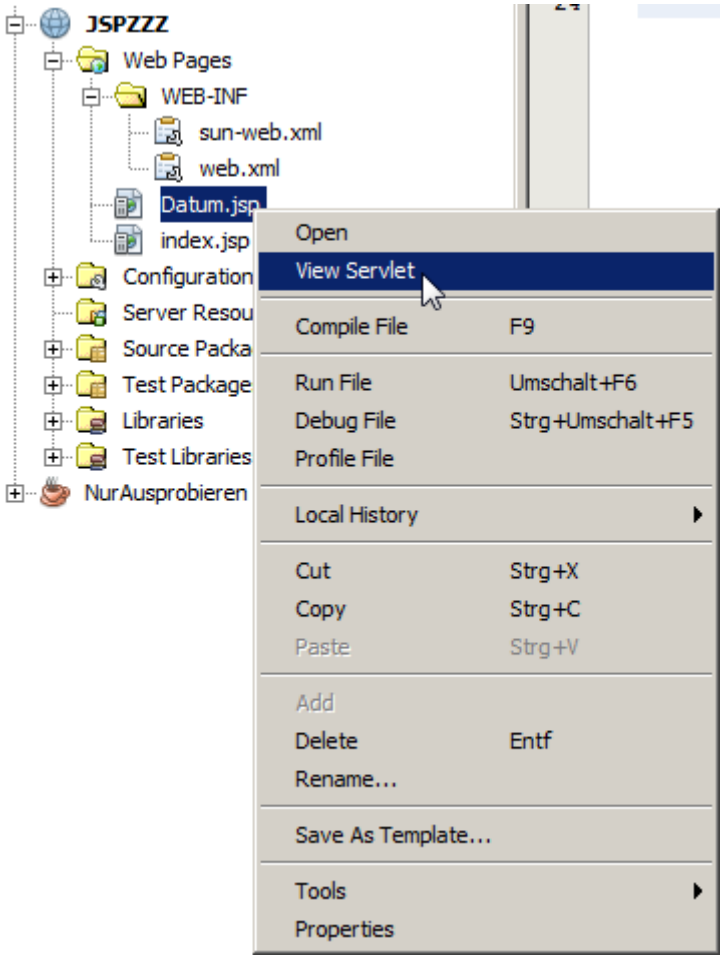
Startet man dann die Applikation erhält man eine Fehlerausgabe im Browser.



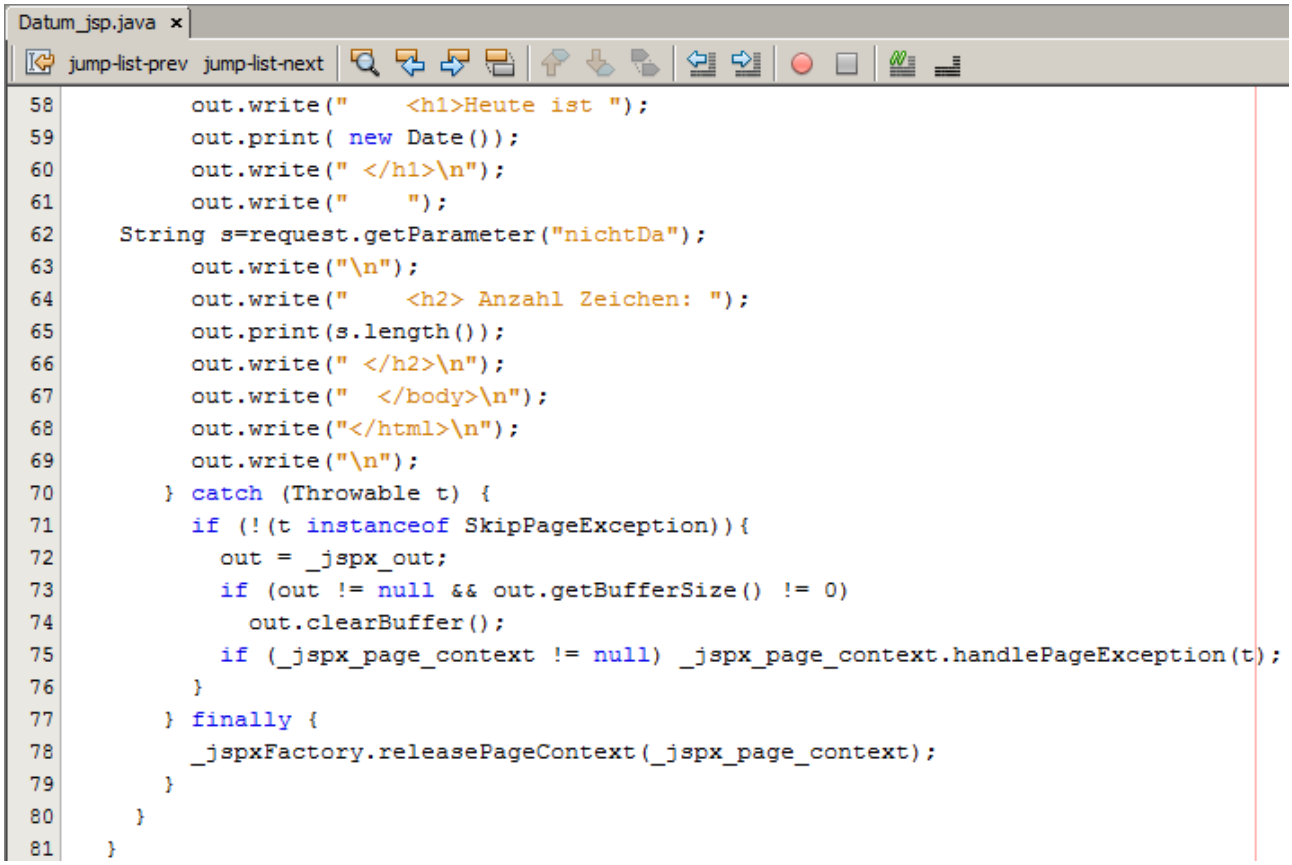
Genauere Informationen zum Fehler stehen im Server-Log.



Der Fehler wird für die generierte Java-Klasse angezeigt. Diese Klasse kann man sich auch in NetBeans anzeigen lassen, dazu wird ein Rechtklick auf der JSP-Seite gemacht und „View Servlet“ ausgewählt.

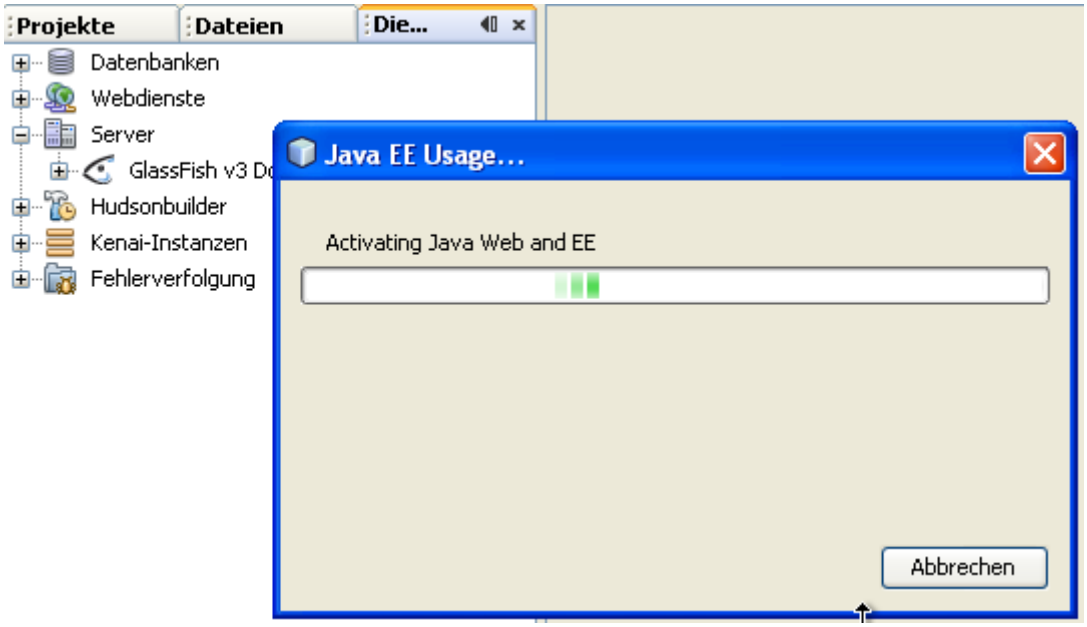


Nun wird der Quelltext des Servlets angezeigt und man kann zur kritischen Zeile, hier 65, gehen und den Fehler analysieren. In diesem Fenster bietet NetBeans keinen weiteren Support an, weiterhin ist es sinnlos, hier Änderungen vorzunehmen, da diese Klasse bei der erneuten JSP-Übersetzung wieder überschrieben wird.

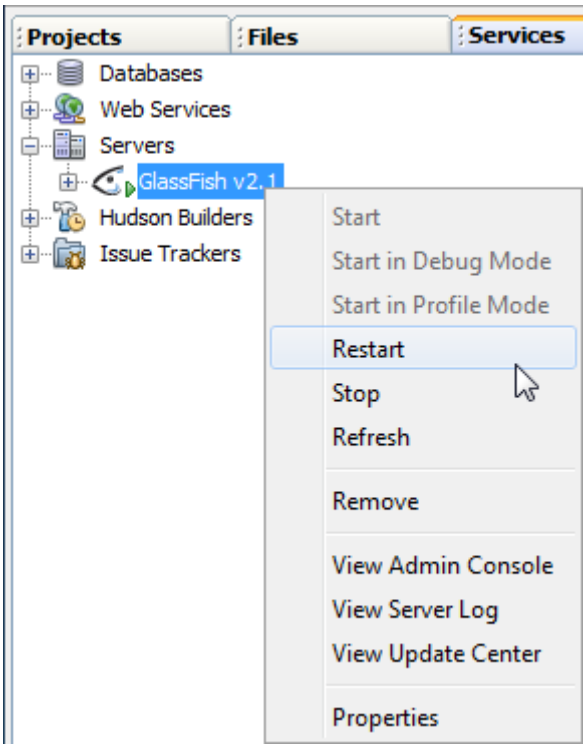


8.4 Informationen zum GlassFish-Server

Den momentanen Status von GlassFish kann man unter dem Reiter Services feststellen. Wird der Server das erste Mal angewählt, wird die zugehörige NetBeans-Funktionalität automatisch aktiviert.

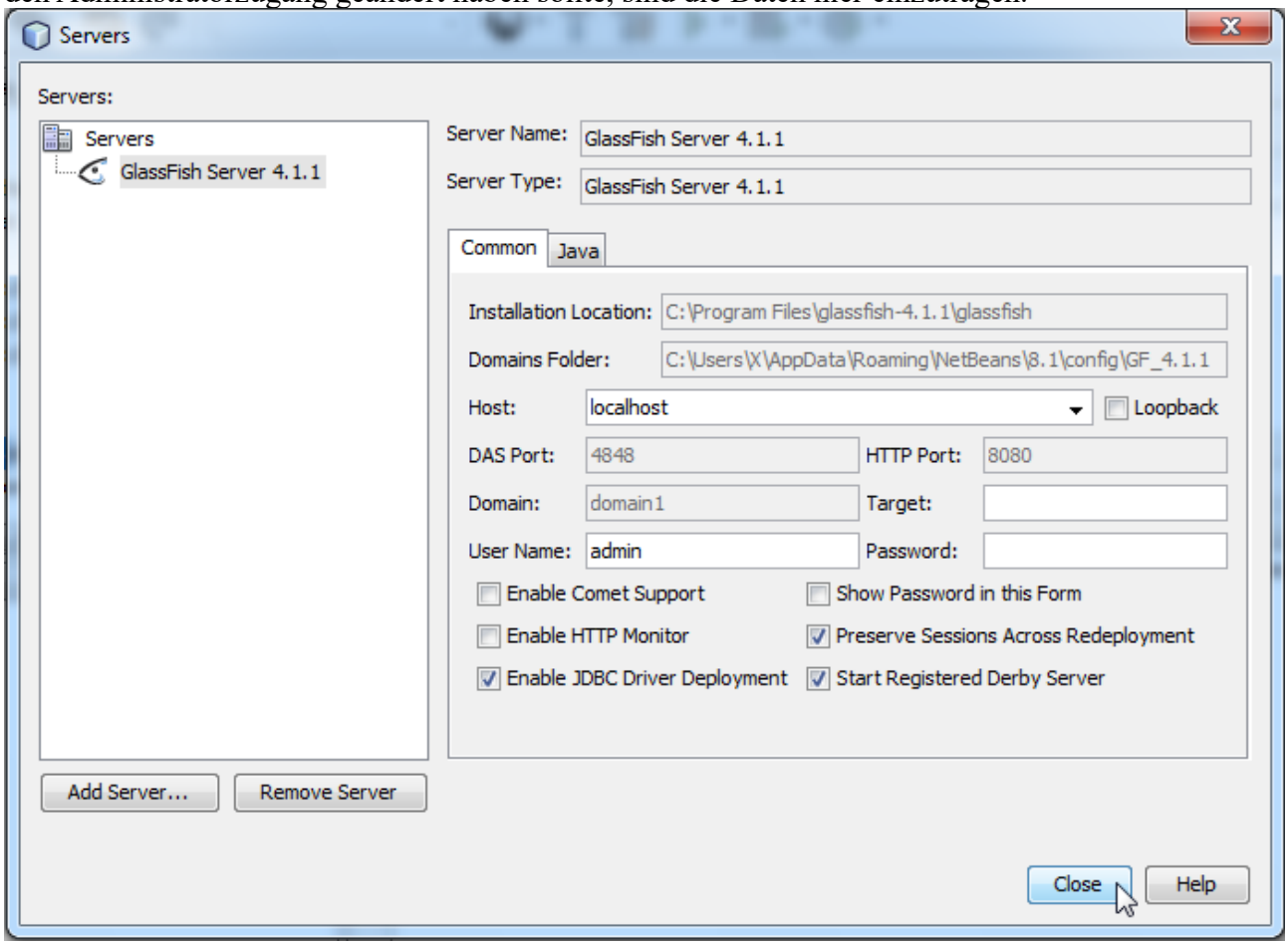


Durch einen Rechtsklick auf den Server erhält man verschiedene Bearbeitungsmöglichkeiten. Im folgenden Fall läuft der Server bereits und könnte jetzt z. B. angehalten werden.

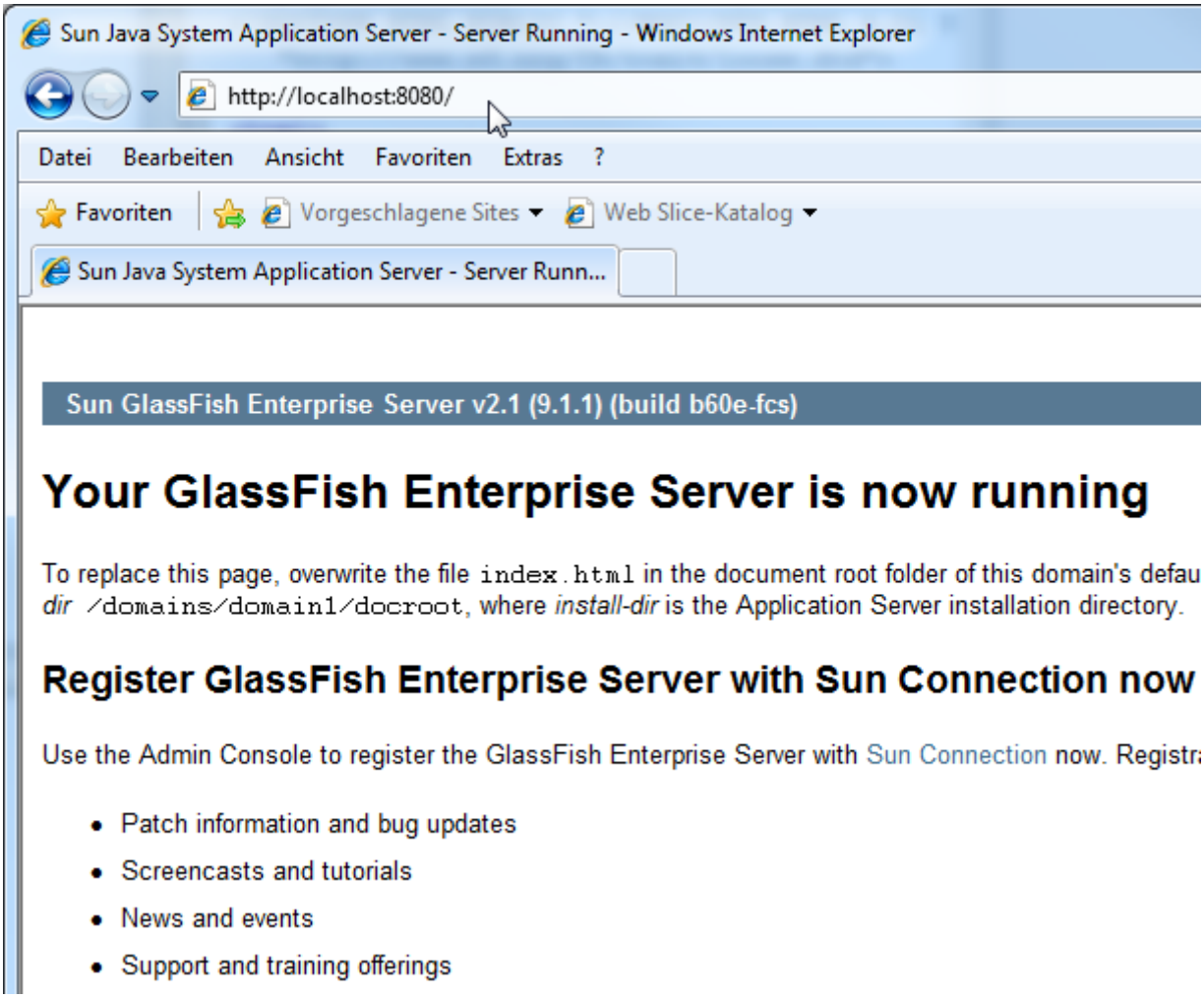


Wählt man den unteren Eintrag Properties, so kann man einstellen, wie sich die Programme beim Versuch der automatischen Installation der Applikationen auf GlassFish dort anmelden. Falls man

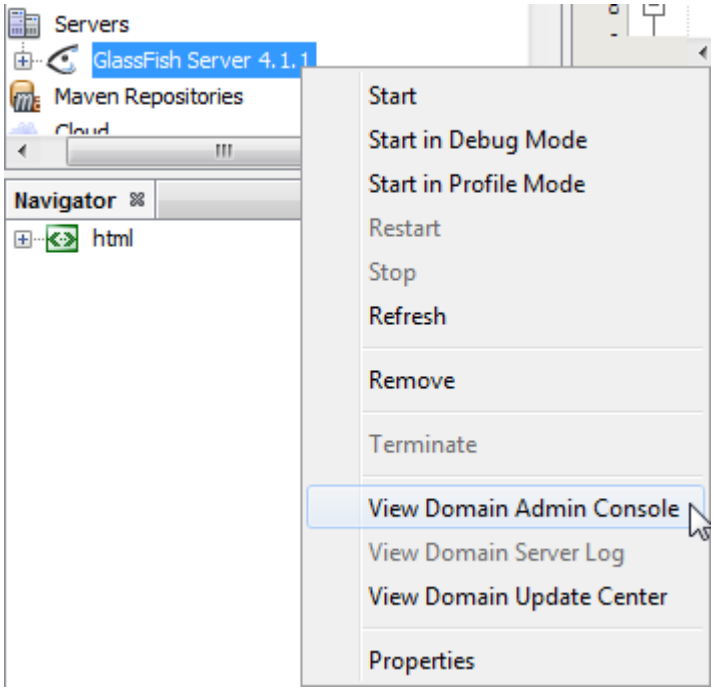
den Administratorzugang geändert haben sollte, sind die Daten hier einzutragen.



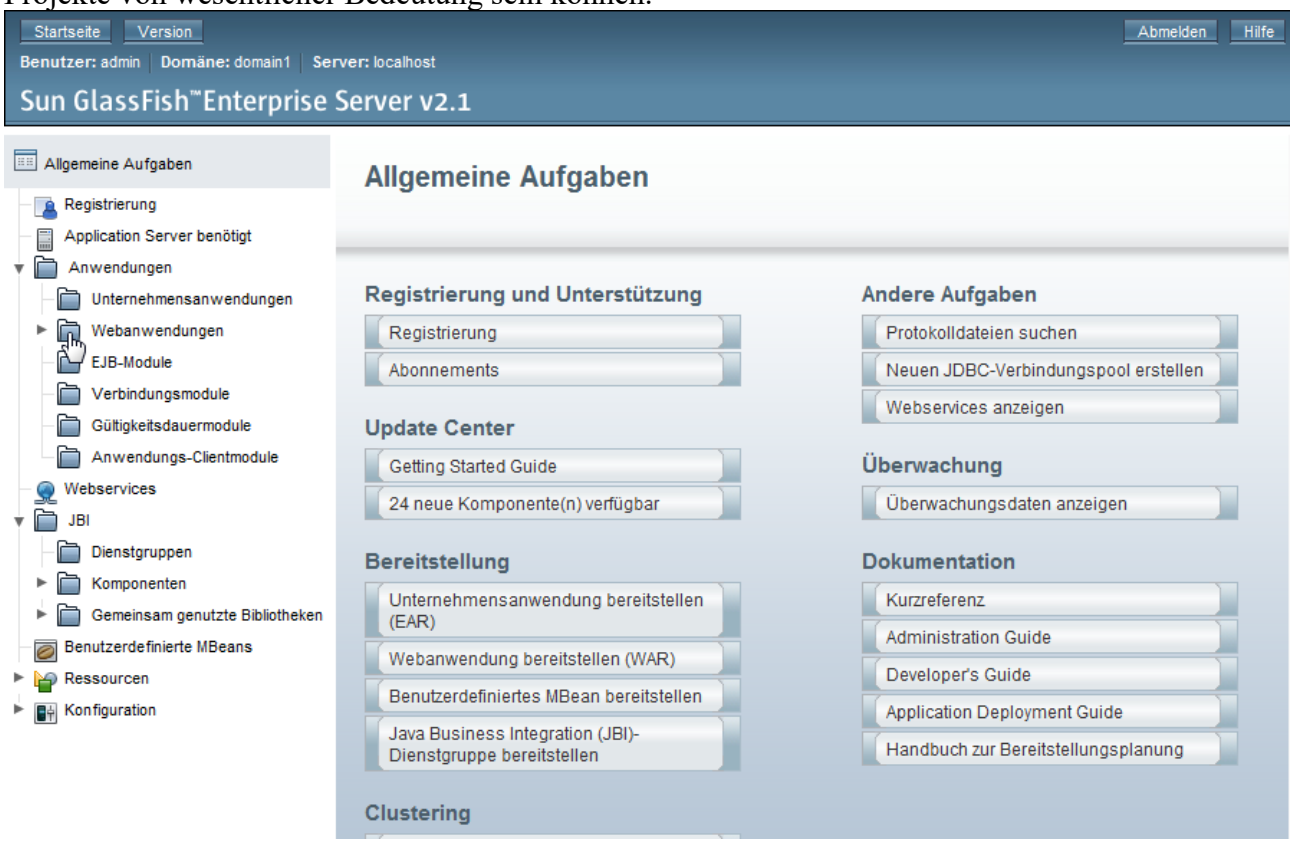
Bei einer Standardinstallation lauscht der Server auf dem Port 8080, was man durch einen Internet-Browser unter der Adresse localhost:8080 feststellen kann.



Die eigentliche Administration findet unter dem Port 4848 statt. Die zugehörige Webseite kann über einen Rechtsklick auf dem Server und „View Domain Admin Console“ aufgerufen werden.



Insgesamt stehen hier sehr mächtige Möglichkeiten zur Verfügung, die für den Erfolg sehr großer Projekte von wesentlicher Bedeutung sein können.

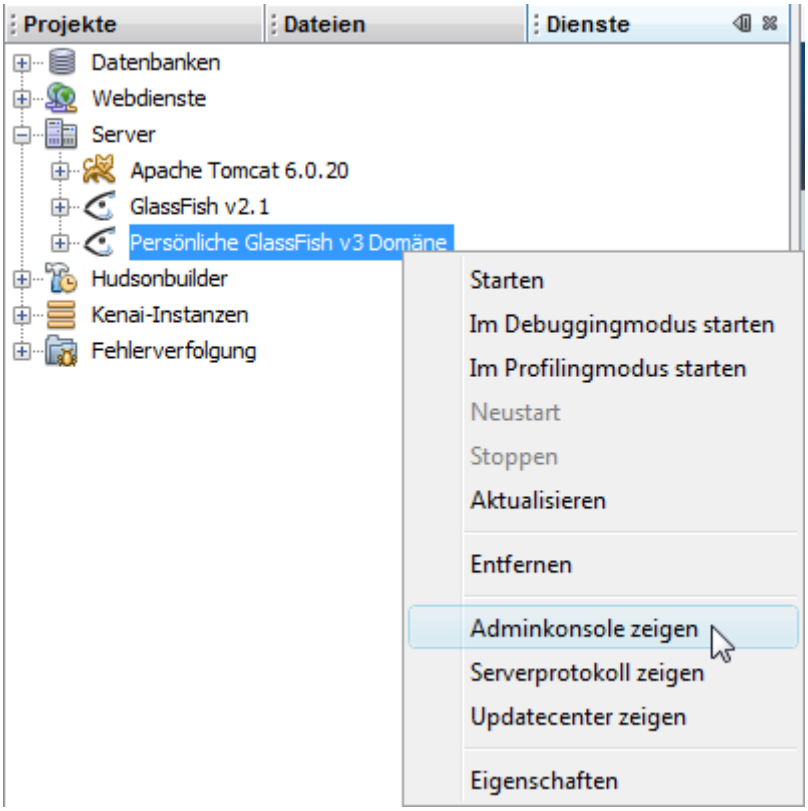


Durch einen Klick auf „Webanwendungen“ auf der linken Seite, findet man z. B. die am Anfang des Kapitels deployte JSP-Seite.

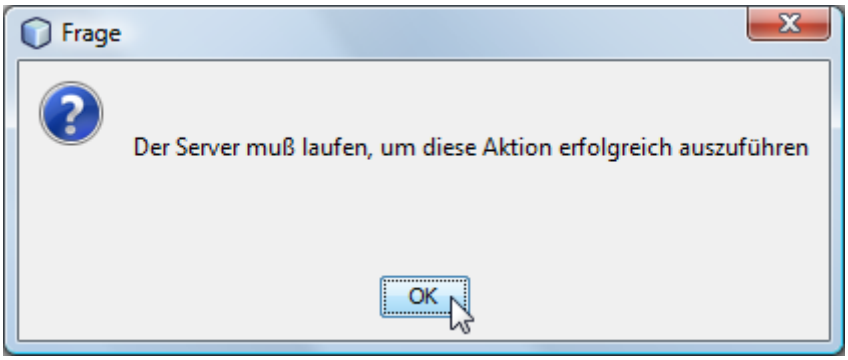


8.5 Nutzereinrichtung in GlassFish

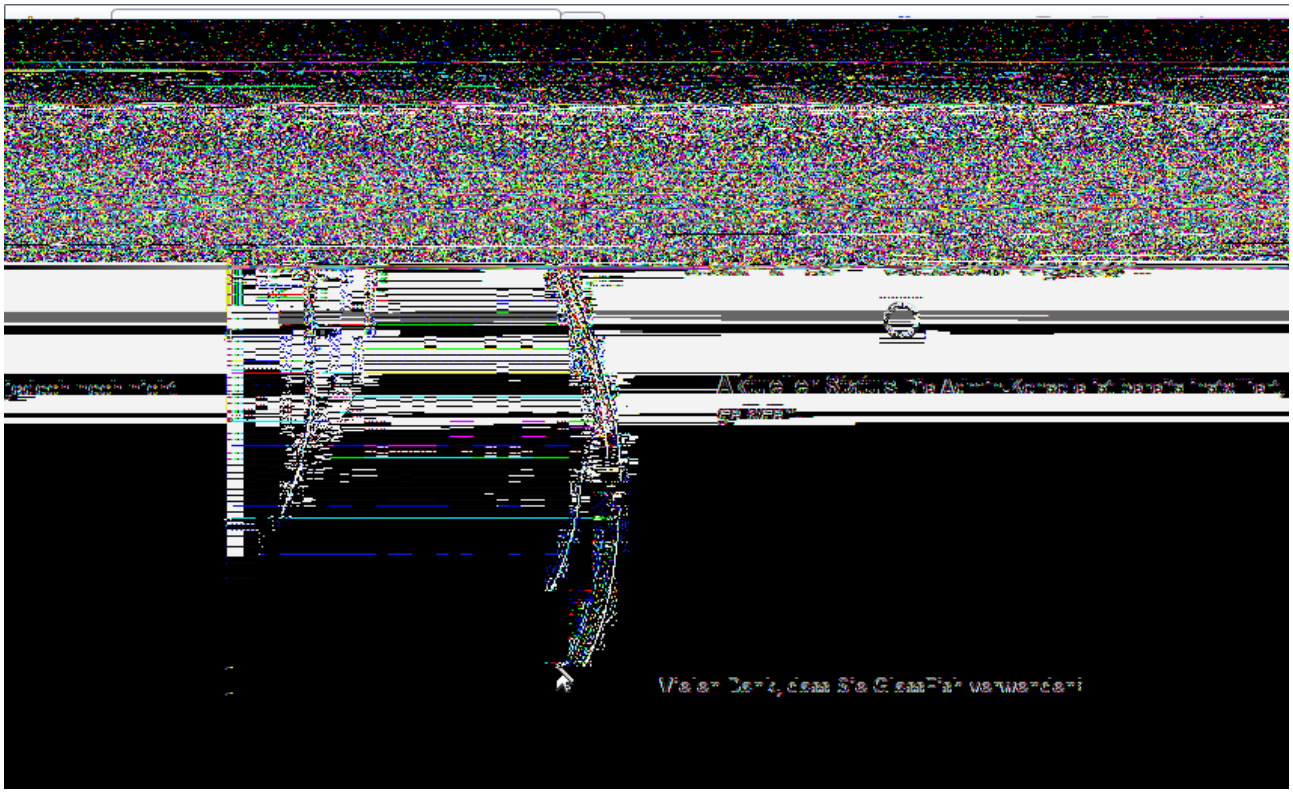
Dieser Abschnitt zeigt die wesentlichen Konzepte, wie man gewisse Servlets, JSP und JSF-Seiten nur bestimmten Nutzern in bestimmten Rollen zugänglich macht. Gerade auf der Serverseite werden nur grundlegende Prinzipien vorgestellt, die für erste Erfahrungen auf diesem Gebiet genügen. Die Einbindung von Tabellen mit Namen, Passwörtern und Rollen sowie die Anbindung eines LDAP-Servers weder hier nicht beschrieben, spielen aber in der Praxis eine große Rolle.
Zunächst werden einige Rollen und Nutzer für den GlassFish-Server eingerichtet, diese Schritte sind für jeden Applikations-Server leider individuell. Die Administrationskonsole ist auch direkt aus NetBeans erreichbar, hierzu wird links oben der Reiter „Dienste“ ausgewählt, eine Rechtsklick auf „Persönliche GlassFish v3 Domäne“ gemacht und „Adminkonsole zeigen“ ausgewählt.



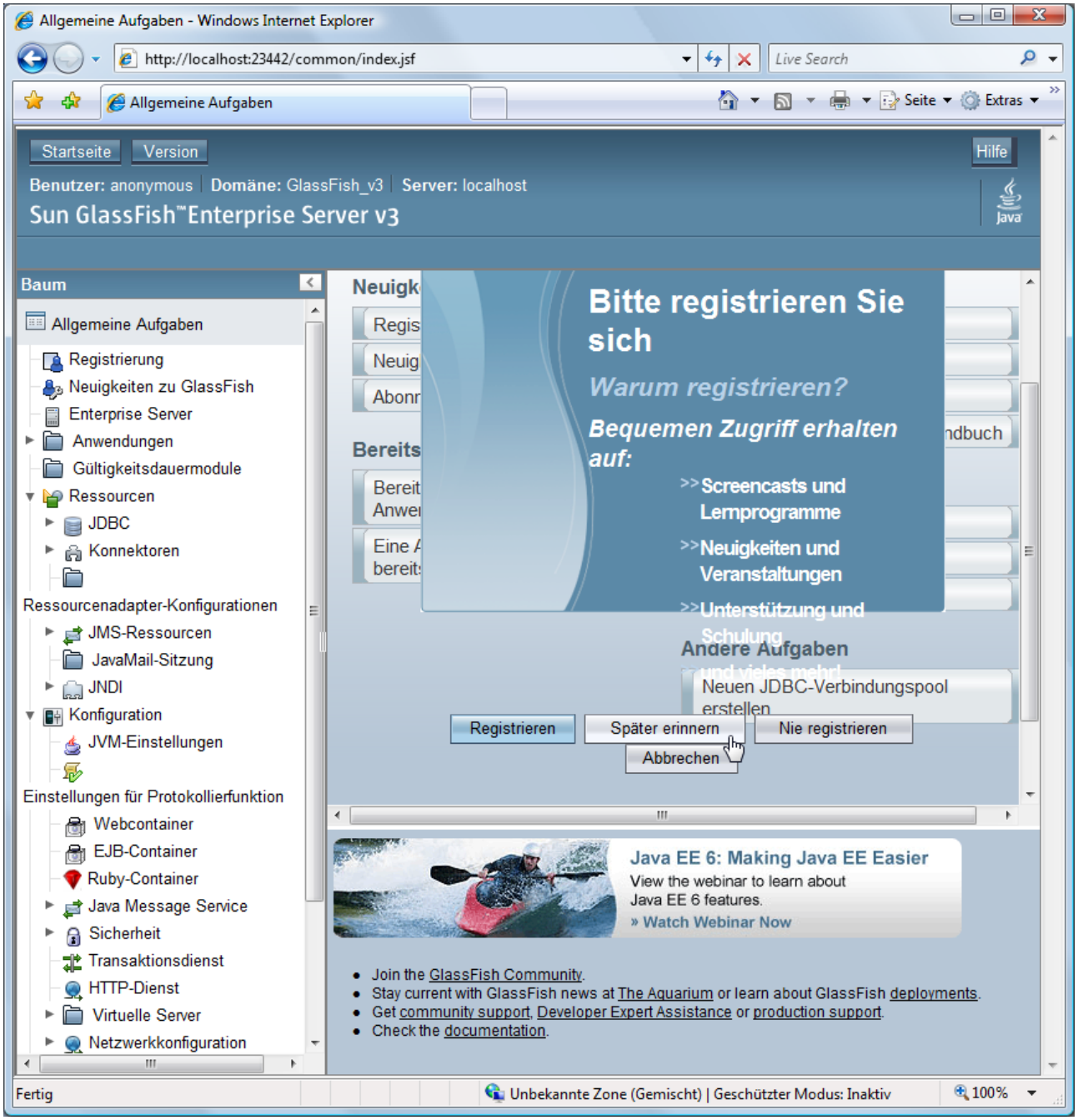
Trotz Rechtschreipfehler sollte man folgenden Rat annehmen und den Server im vorher gezeigten Menü starten, um dann zur Adminkonsole zu wechseln.



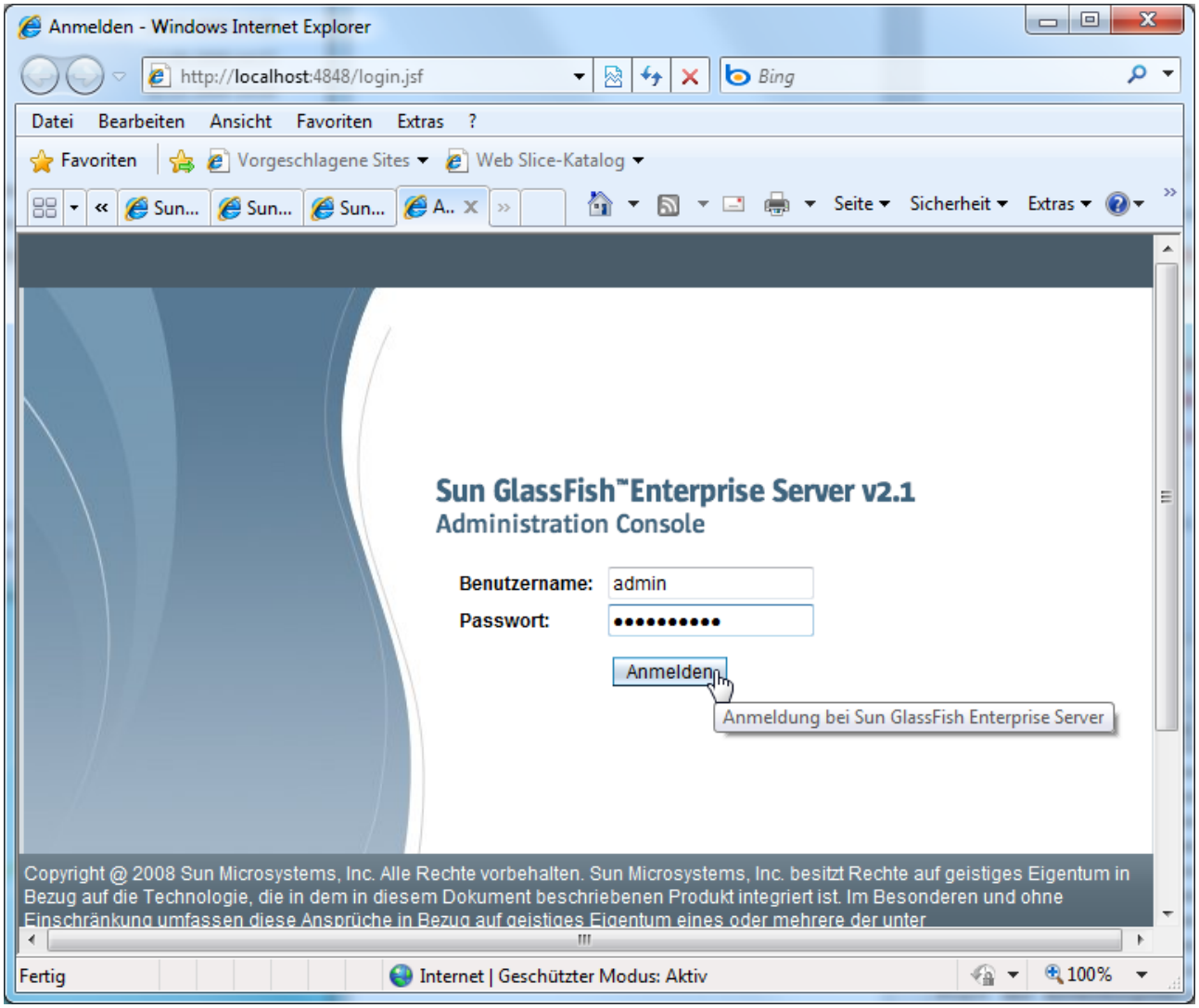
Der erste Start des Servers und der Aufruf der Administration kann etwas dauern.



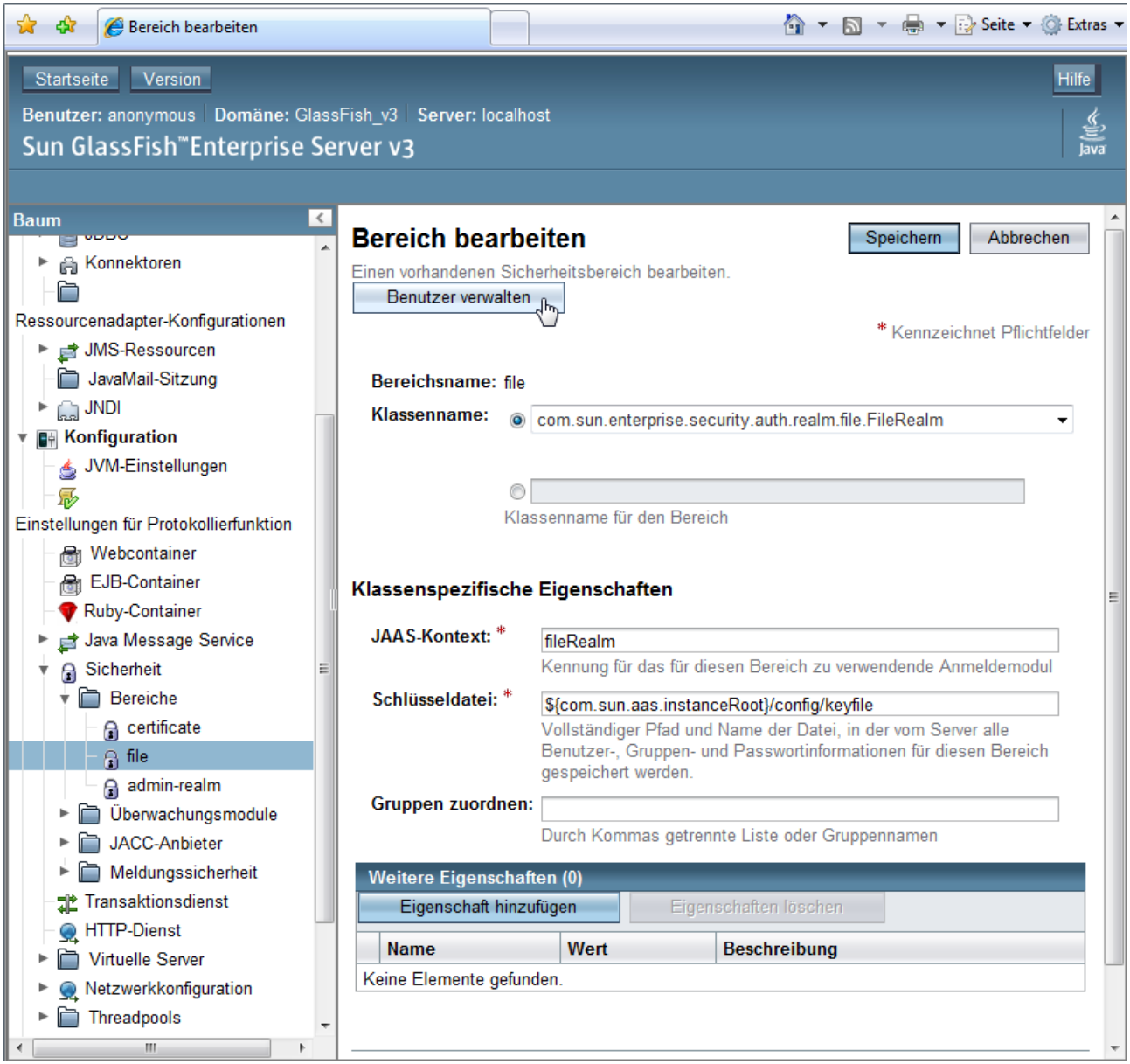
Gegebenenfalls bekommt man zunächst eine Aufforderung sich zu registrieren, was kostenlos ist, aber nicht gemacht werden muss.



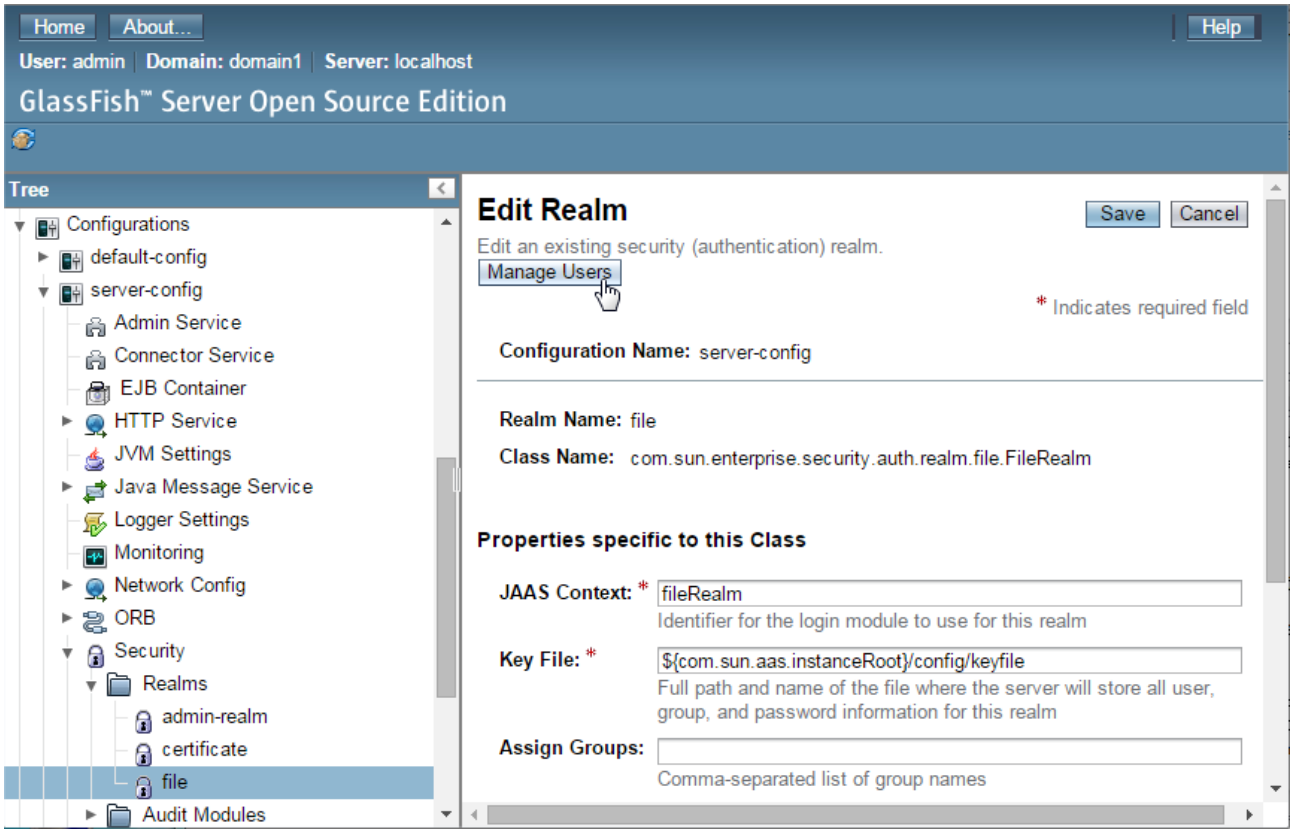
Da es sich um eine persönliche Domain handelt, also anschaulich einem privaten Server, muss man sich nicht explizit anmelden. Bei GlassFish V2 müssen Nutzer („admin“) und Passwort („adminadmin“) eingegeben werden.



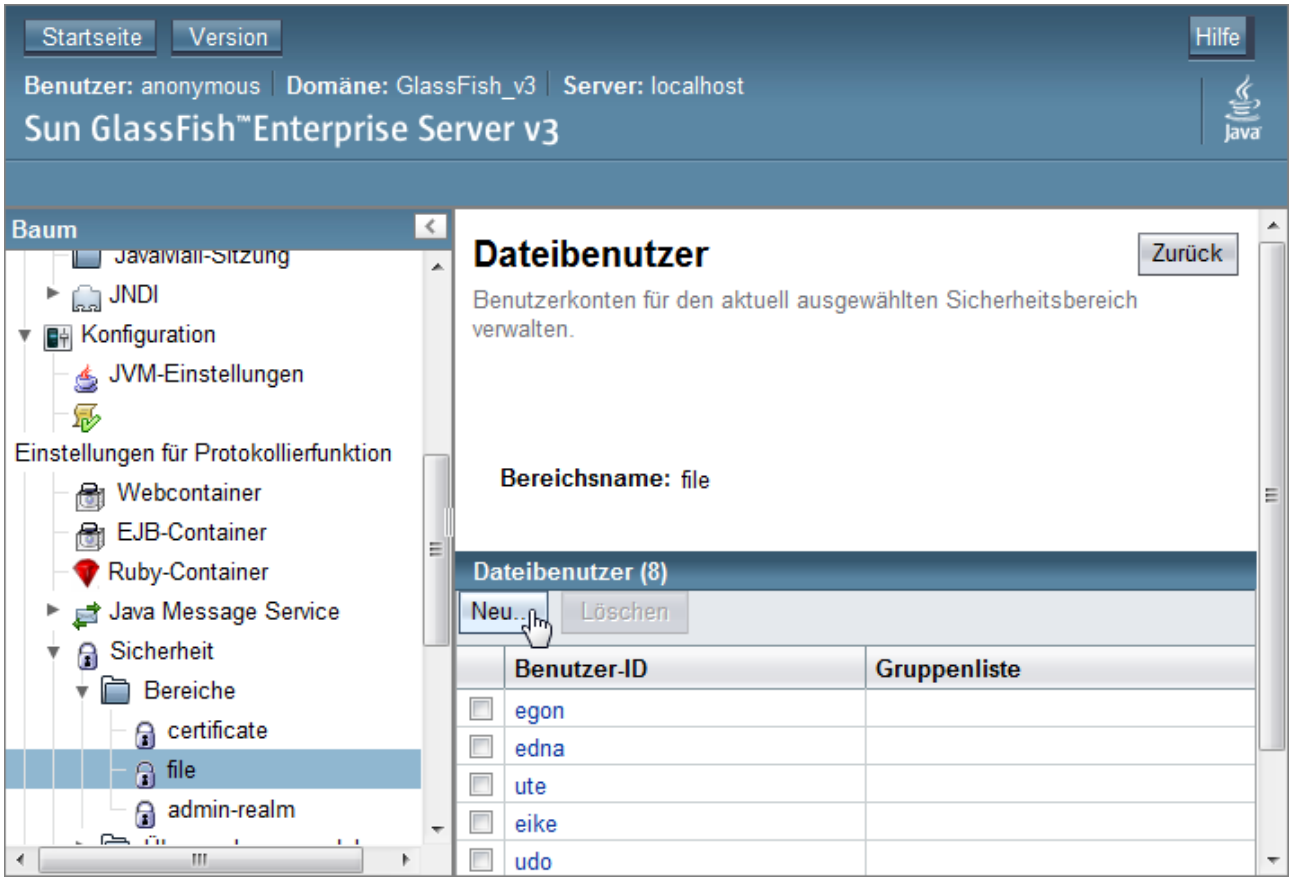
Nun wird im Browser auf der linken Seite zu „Konfiguration -> Sicherheit -> Bereiche -> file“ manövriert und oben auf „Benutzer verwalten“ geklickt.



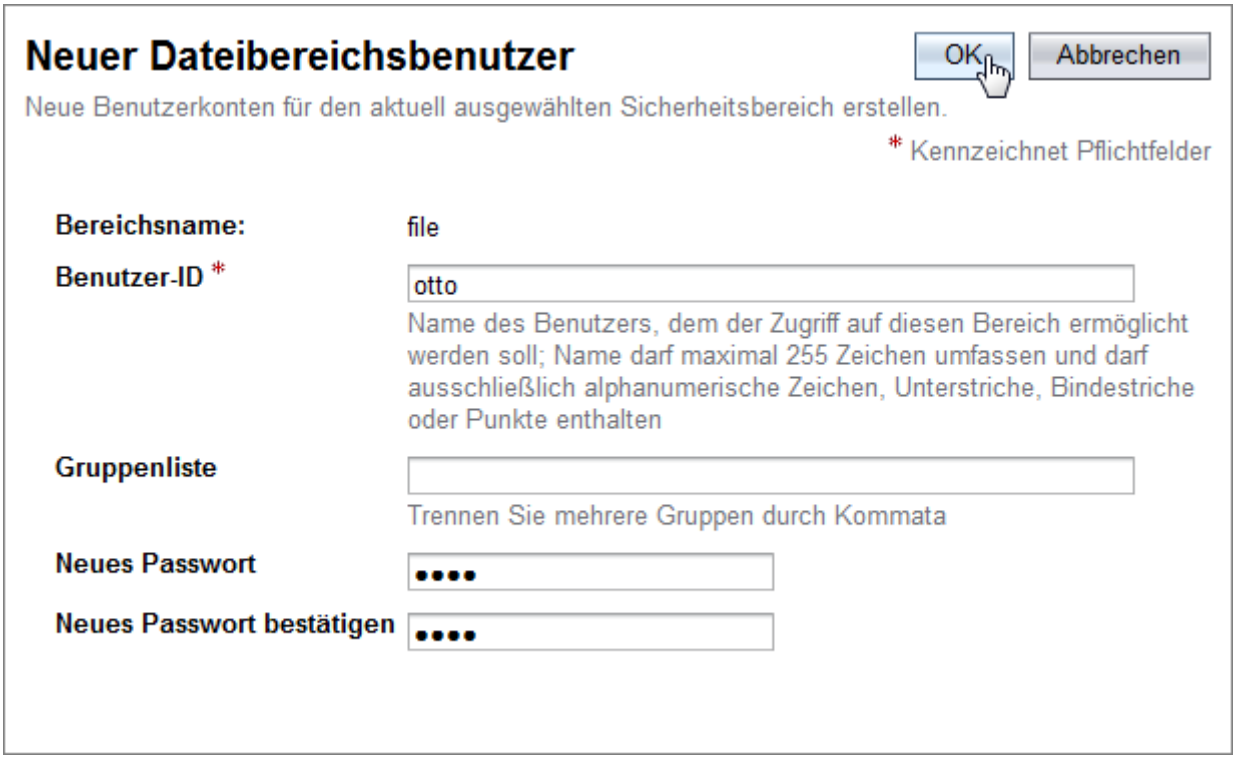
In Englisch unter „Configurations > server-config > Security > Realms > file“:



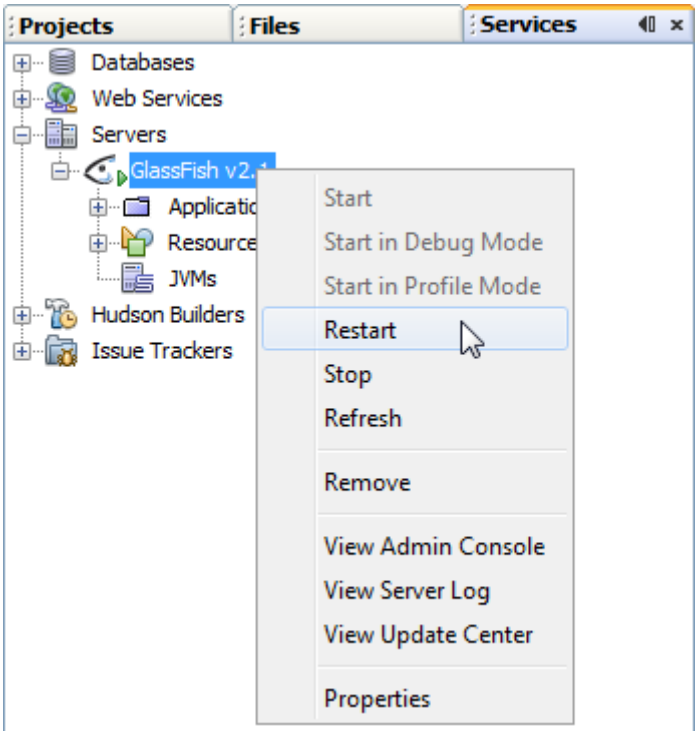
Man erhält eine Übersicht über die vorhandenen Nutzer, die im Beispiel bereits angelegt wurden. Weiterhin kann über den Knopf „Neu...“ ein neuer Nutzer angelegt werden.



Nun wird unter „Benutzer-ID *“ ein Nutzername angegeben, unter Gruppenliste ist eine Liste dem Nutzer zugeordneter Gruppen ergänzbar (hier fehlt leider eine Auswahlunterstützung oder die einfache Einrichtung von Rollen). Unten wird zweimal das gleiche neue Passwort eingegeben.

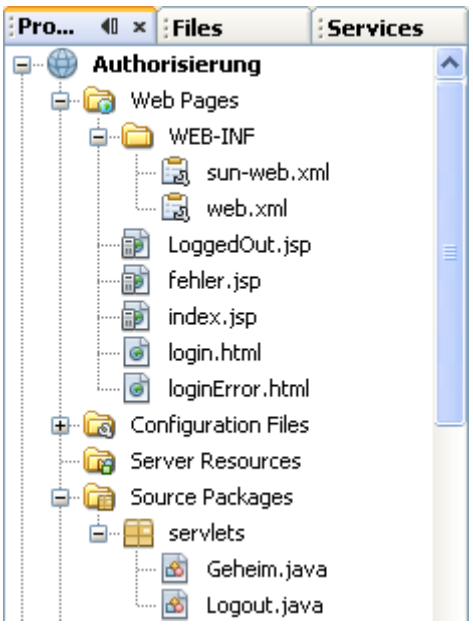


Generell ist nach Änderungen der Server-Einstellungen der Server neu zu starten. Dies ist auch direkt in NetBeans möglich.

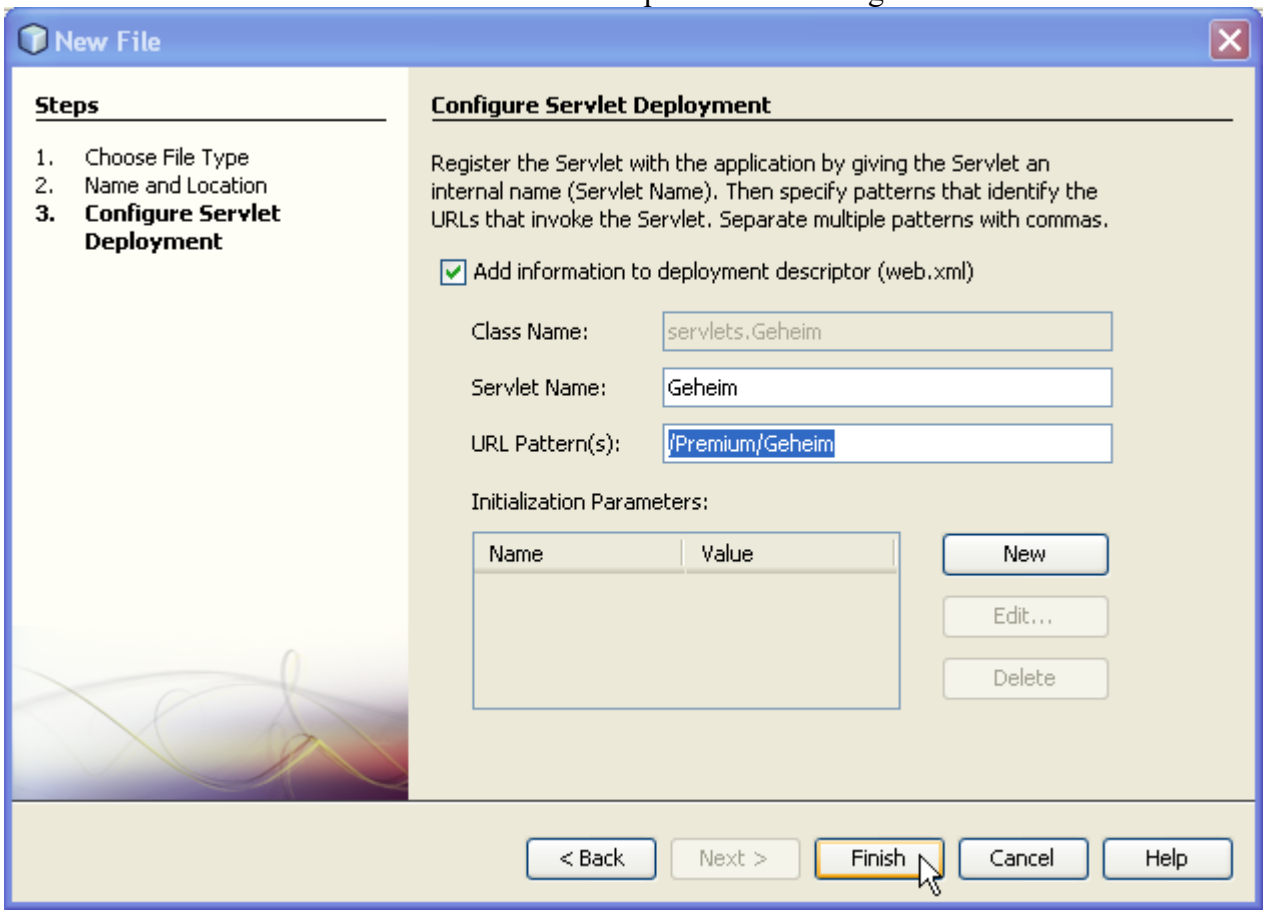


8.6 Sichere JSP-Seiten

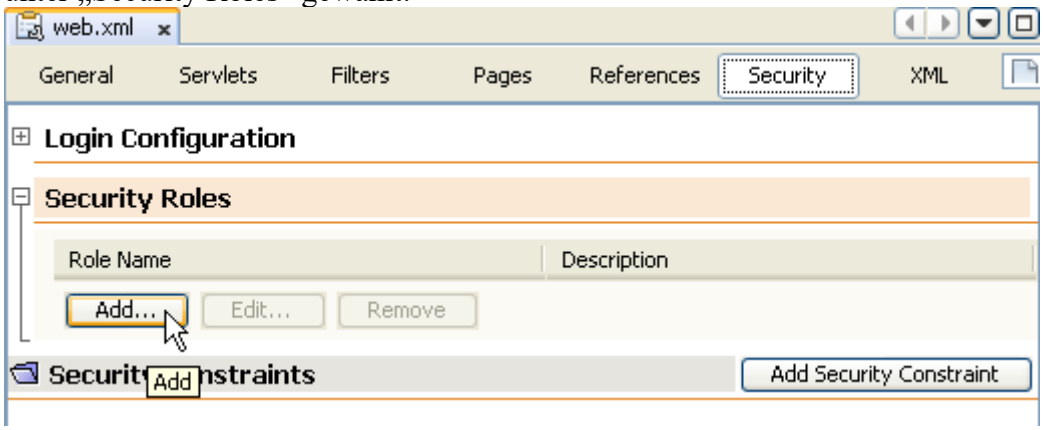
Für das konkrete Beispiel wird davon ausgegangen, dass es ein schützenswertes Servlet Geheim.java gibt, das über die Startseite index.jsp angesprungen werden kann. Die Projektstruktur sieht wie folgt aus.



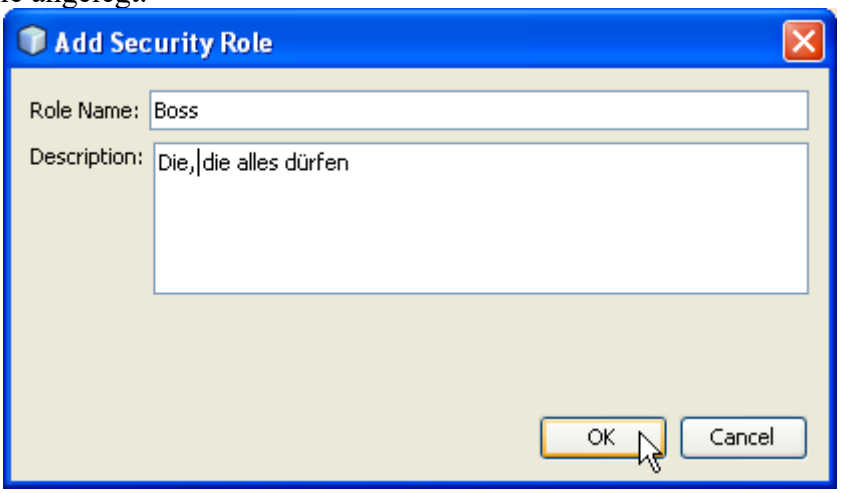
Beim Anlegen von zu schützenden Seiten ist es sinnvoll, diese in logischen Unterordnern zu organisieren und später diesen Ordnern bestimmten Nutzer(-gruppen) zuzuordnen. Bei der Servlet-Entwicklung kann man z. B. direkt beim „URL Pattern“ die Einordnung in ein logisches Unterverzeichnis erreichen. Alternativ kann dies später in web.xml geändert werden.



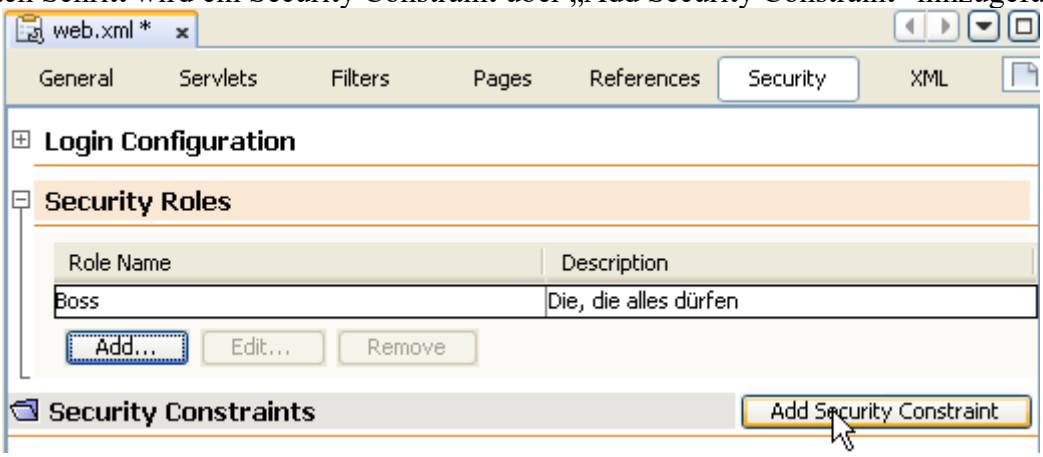
Nun wird im Deployment Descriptor festgelegt, dass nur ein Nutzer mit der Rolle „Boss“ auf das geschützte Servlet zugreifen darf. Dazu wird web.xml geöffnet, der Reiter „Security“ gewählt und „Add…“ unter „Security Roles“ gewählt.



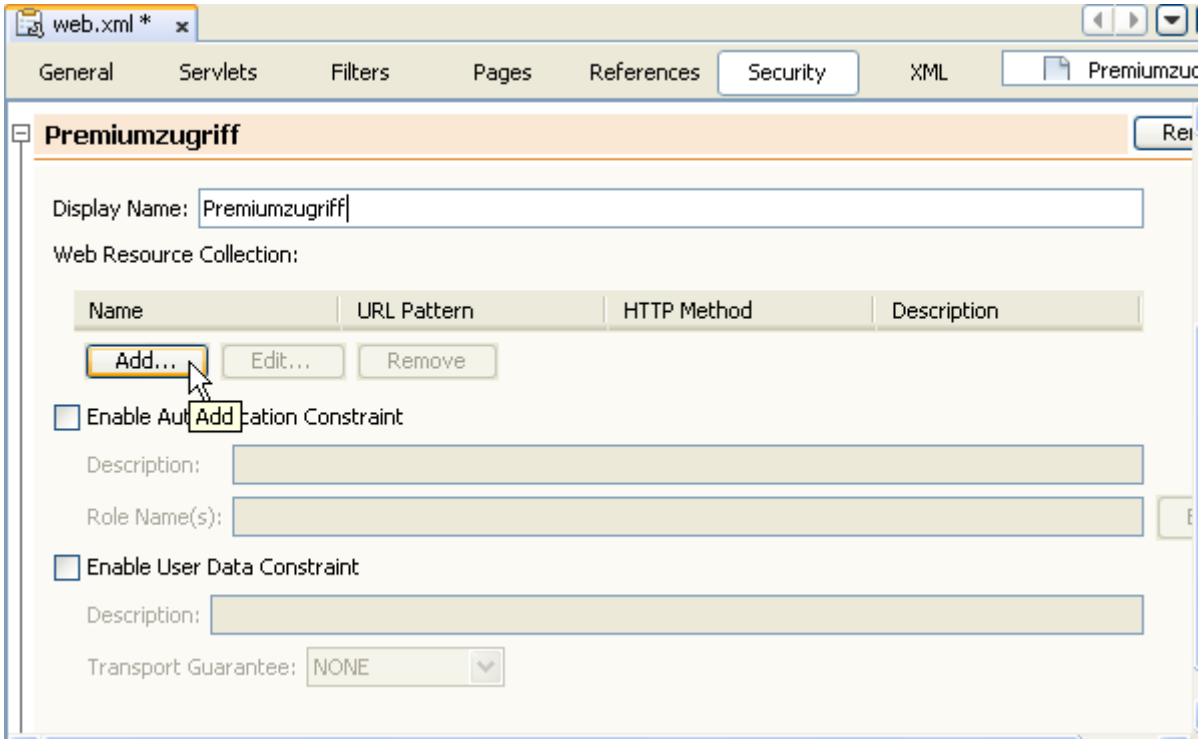
Nun wird die Rolle angelegt.



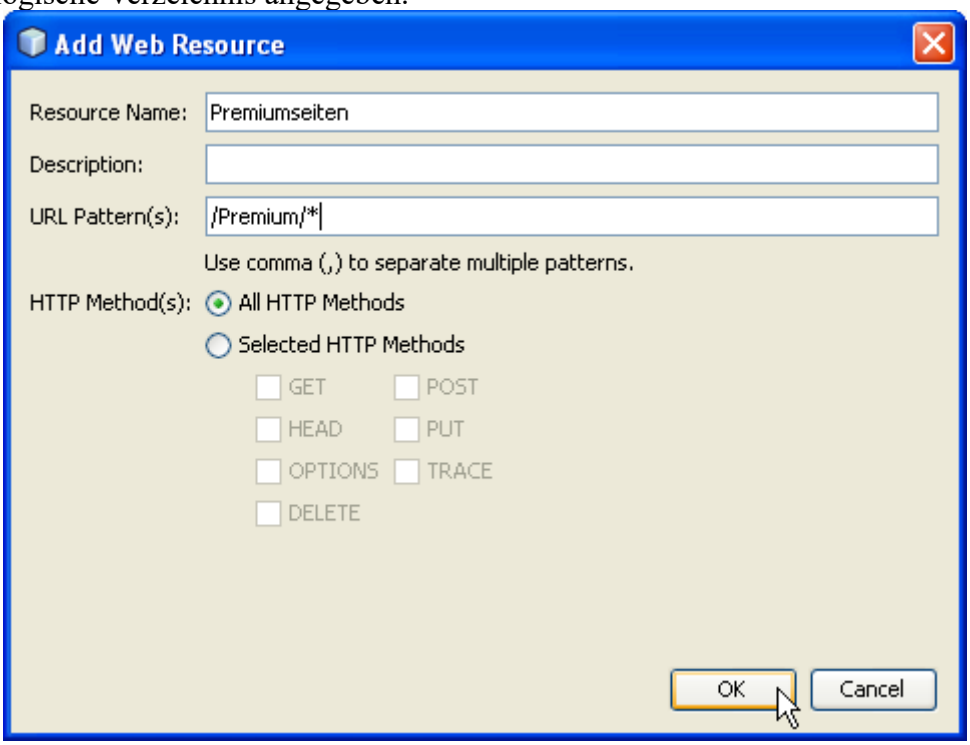
Im nächsten Schritt wird ein Security Constraint über „Add Security Constraint“ hinzugefügt.



Hier kann zunächst ein Name eingegeben werden. Danach wird über „Add…“ angegeben, welche Ressourcen zu schützen sind.

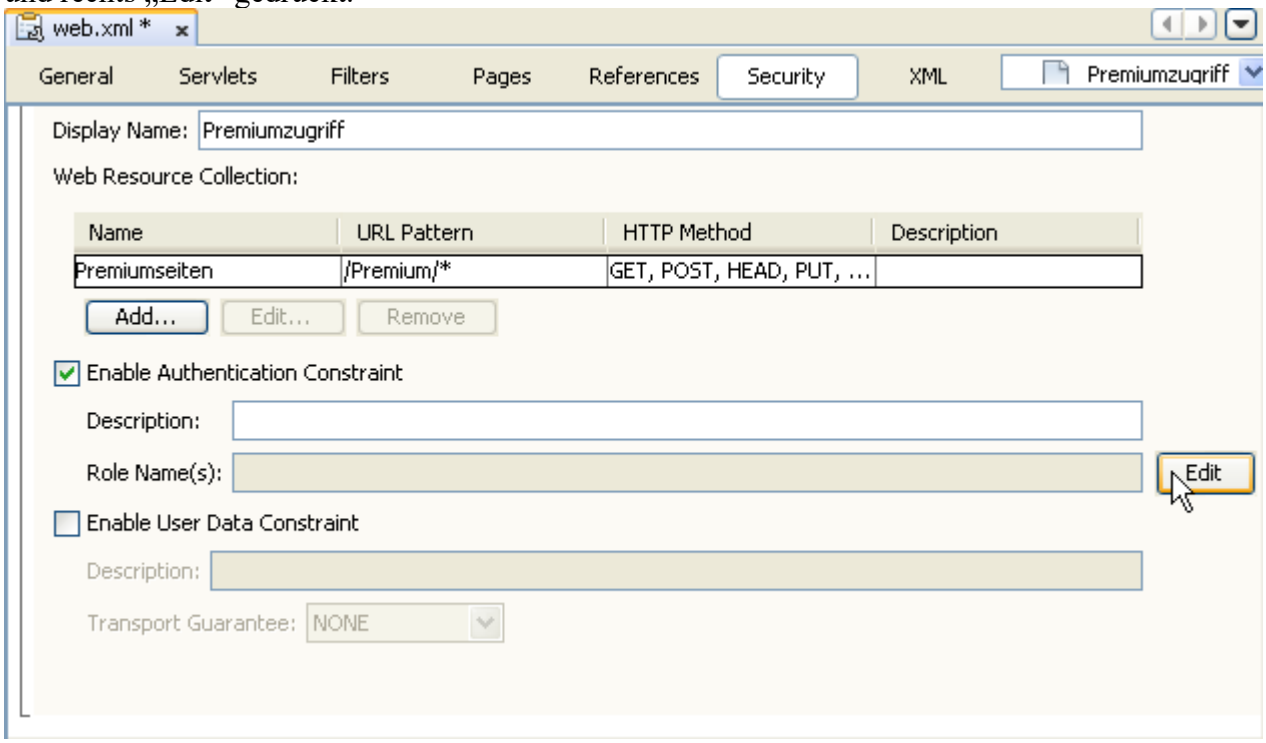


Der angegebene Name erleichtert die Identifizierung. Beim „URL-Pattern“ wird hier das zu schützende logische Verzeichnis angegeben.

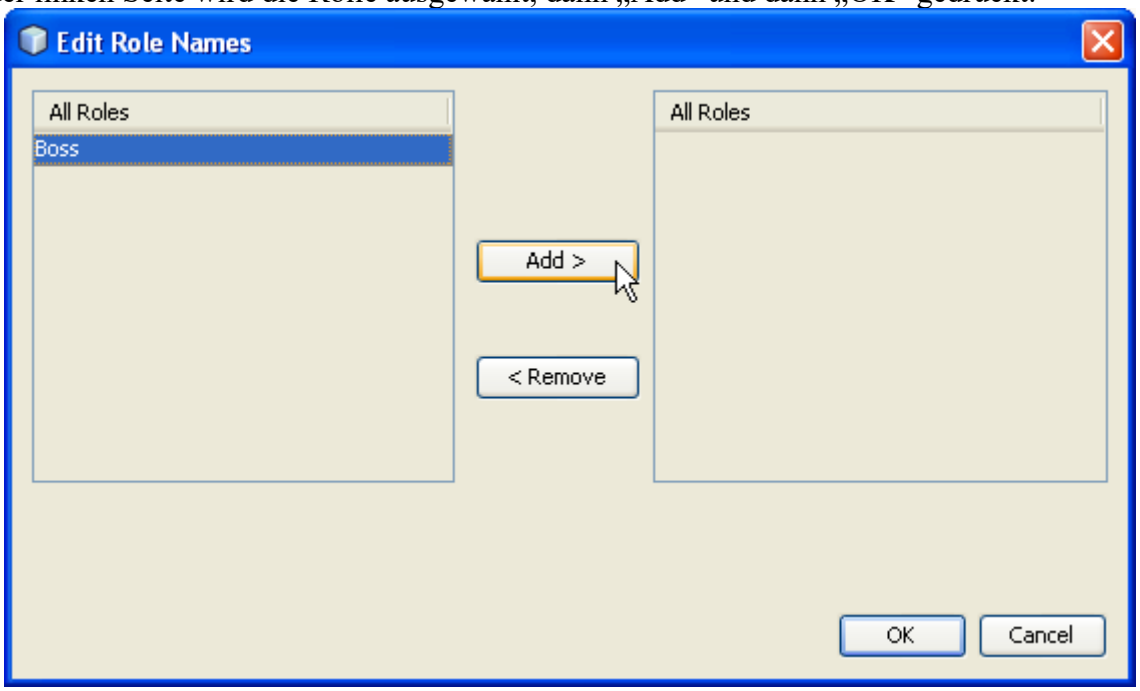


Nun wird die zugehörige Rolle angegeben, dazu wird „Enable Authentication Constraint“ markiert

und rechts „Edit“ gedrückt.

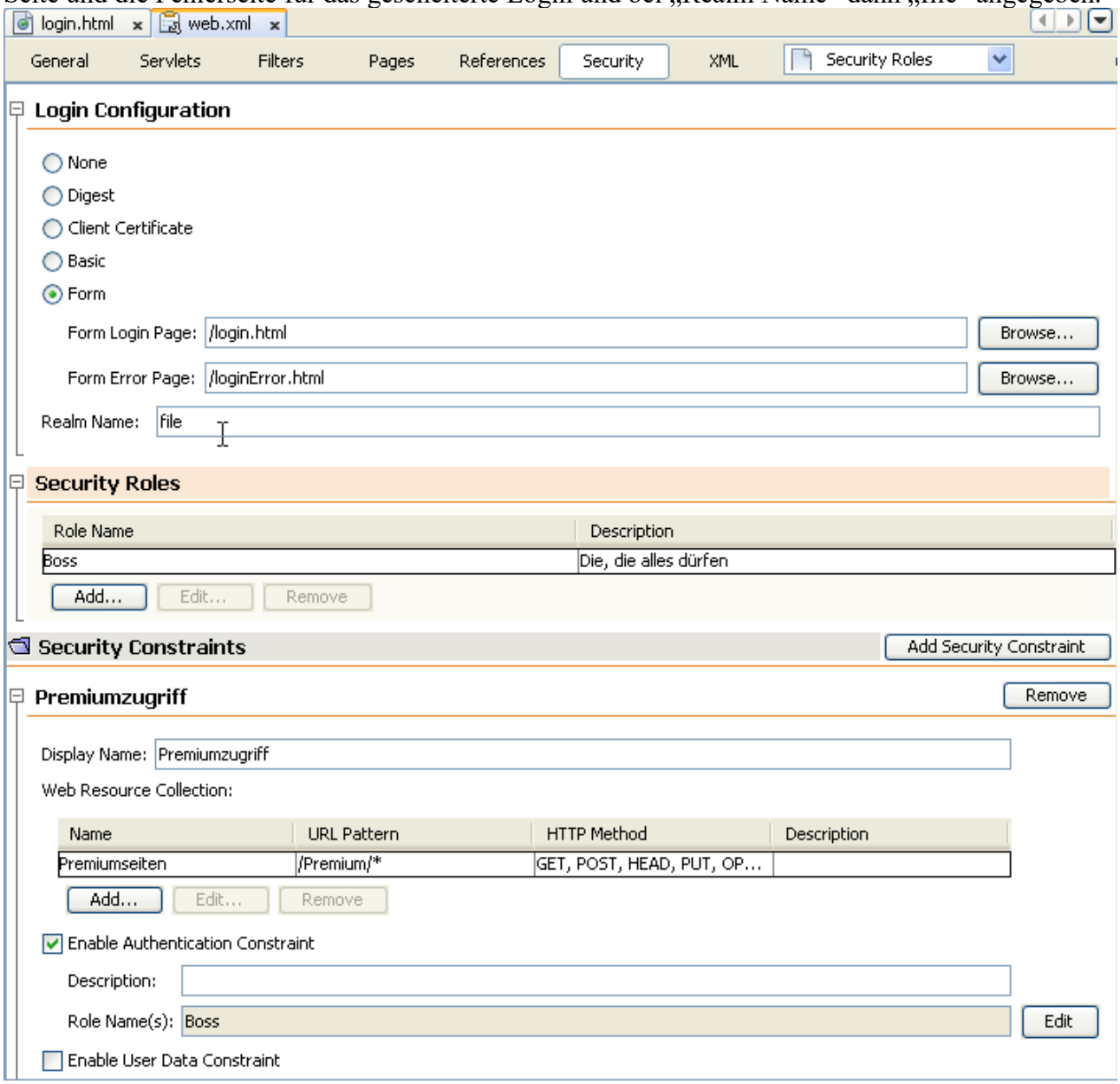


Auf der linken Seite wird die Rolle ausgewählt, dann „Add“ und dann „OK“ gedrückt.



Generell gibt es verschiedene Absicherungsmöglichkeiten. Die hier vorgestellte nutzt eine eigene Login-Seite, die noch weiter abgesichert werden müsste, da die Informationen sonst im Klartext übertragen würden. Hier wird zunächst der einfache Ablauf beschrieben, um danach auf dieses

Problem einzugehen und mit https zu lösen. Bei den Einstellungen wird „form“ gewählt, die Login-Seite und die Fehlerseite für das gescheiterte Login und bei „Realm Name“ dann „file“ angegeben.



Die Datei login.html hat folgenden Inhalt, dabei müssen die „action“- und „name“-Einträge genau den angegebenen Inhalt haben. Der Typ des Passwortfeldes würde real natürlich auf „password“ gesetzt.

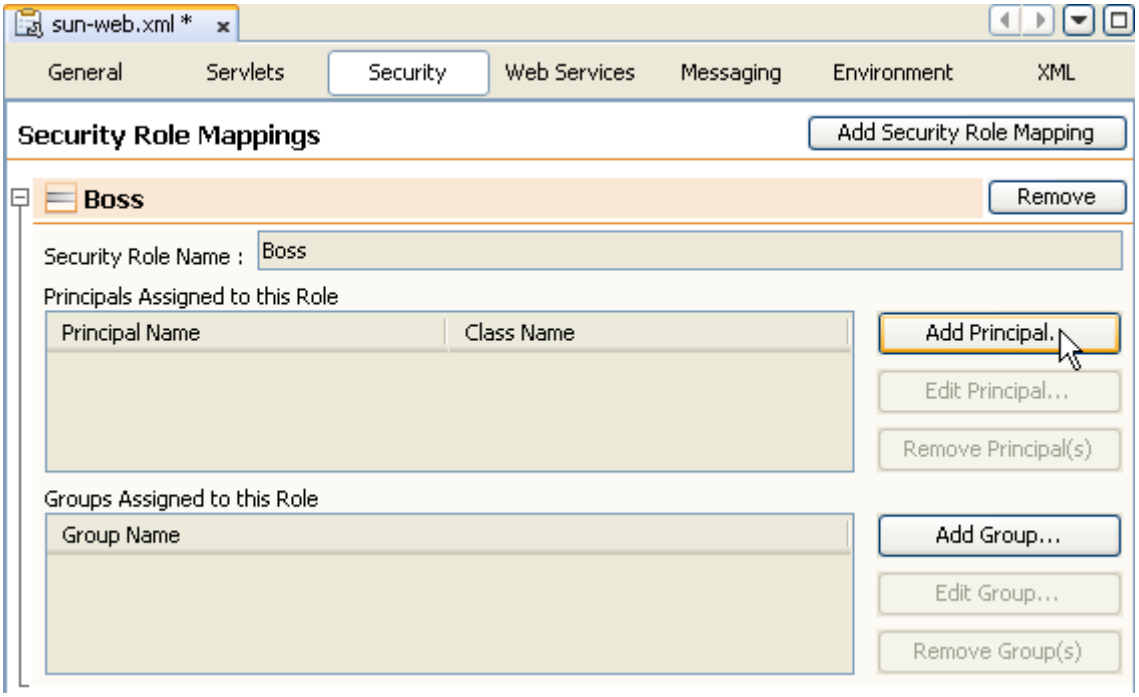
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
```

```
  <body>
      <form name="POST" action="j_security_check">
          Nutzer:<input type="text" name="j_username" value="" size="15" /><br>
          Passwort:<input type="text" name="j_password" value="" size="15" /><br>
          <input type="submit" value="Anmelden" />
      </form>
  </body>
</html>
```

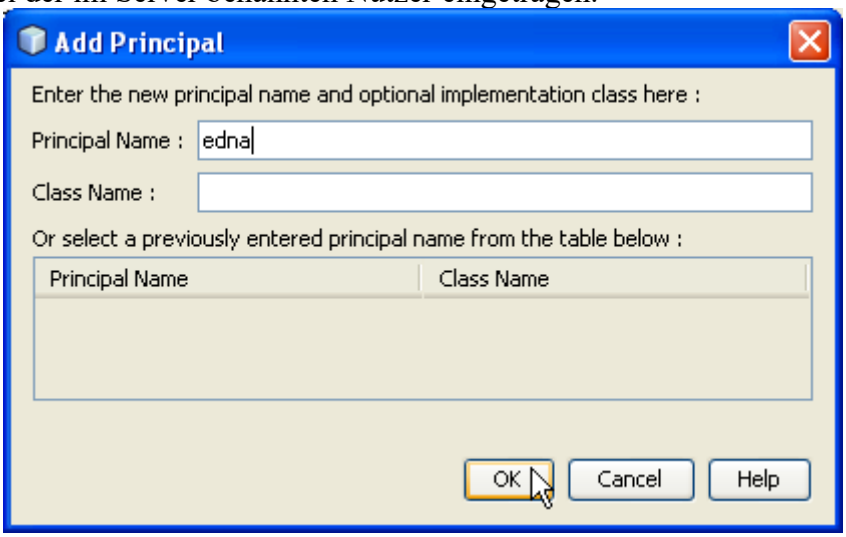
Die Fehlerseite loginError.html hat folgenden Inhalt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Fehler</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    Passt nicht <br><br>
    <a href="login.html">noch'n Versuch</a>
  </body>
</html>
```

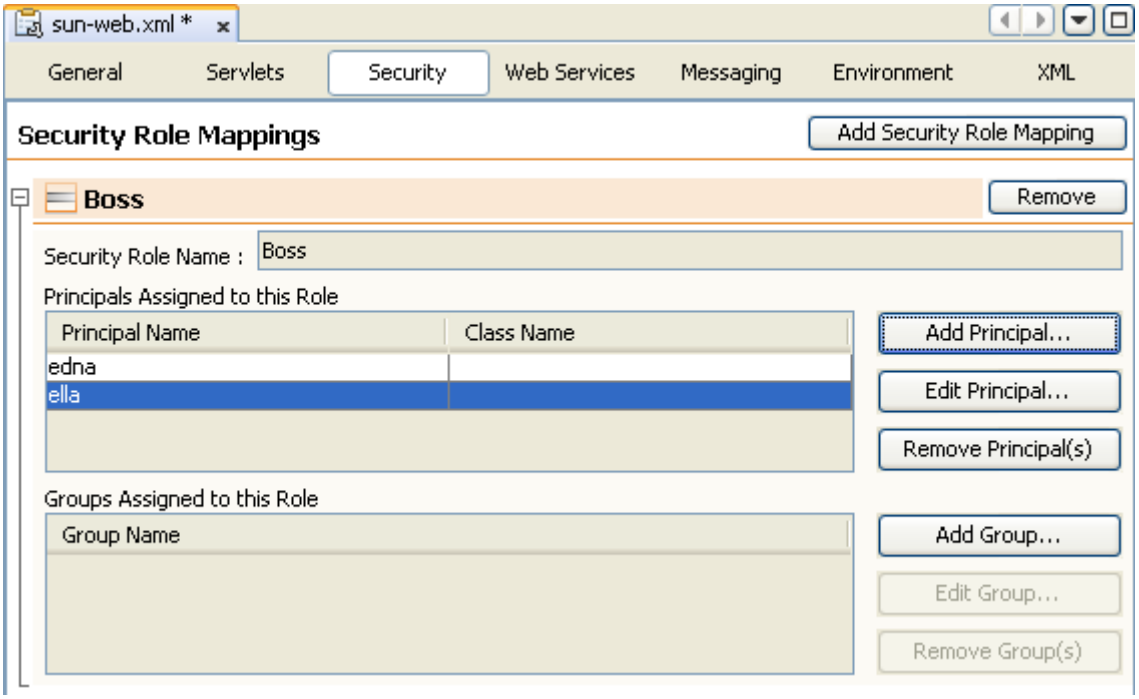
Für die Applikation gibt es jetzt Rollen, die mit den Rollen (Gruppen) im Server verknüpft werden müssen. Eine direkte Übernahme der Rollen vom Server ist denkbar, der genutzte Ansatz ist aber wesentlich flexibler. Die jetzt vorzunehmenden Einstellungen sind Server-spezifisch. Es wird die Datei sun-web.xml geöffnet und der Reiter „Security“ gewählt. Unter „Boss“ wird dann „Add Principal“ angeklickt, wodurch konkrete Nutzer der Rolle zugeordnet werden (die Zuordnung der zum Bereich gehörenden Gruppe wäre auch möglich).



Hier werden zwei der im Server benannten Nutzer eingetragen.



Die Eintragungen für das Beispiel sehen wie folgt aus.



Die Startseite index.jsp hat folgenden Inhalt. Das Session-Attribut IN wird in der Applikation genutzt, um eine verlassene Session nicht wieder mit dem Back-Knopf des Browsers betreten zu können.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Start</title>
    </head>
    <body>
        <form name="m" action="Premium/Geheim">
            <input type="submit" value="Premium" name="mb" />
        </form>
        <c:if test="${IN!=null}">
            <form name="l" action="Logout">
                <input type="submit" value="Ausloggen" name="wb" />
            </form>
        </c:if>
    </body>
</html>
```

Zur Vollständigkeit werden die weiteren Servlets mit zugehörigen JSP-Seiten angegeben. Das Servlet Geheim.java sieht wie folgt aus.

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
```

```
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        request.getSession(false).setAttribute("IN", 0);
        try {
            out.println("Geld stinkt<br><br>");
            out.println("<a href=\"../index.jsp\">zum Start </a>");
        } finally {
            out.close();
        }
    }
```

Das Servlet Logout.java hat im Wesentlichen folgenden Inhalt.

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try {
            if (request.getSession(false) != null) {
                request.getSession().removeAttribute("IN");
                request.getSession().invalidate();
            }
        } finally {
            RequestDispatcher view = request.getRequestDispatcher("LoggedOut.jsp");
            view.forward(request, response);
        }
    }
```

Die zugehörige JSP-Seite sieht wie folgt aus.

```
<%@page contentType="text/html" session="false" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Logged Out</title>
    </head>
    <body>
      <a href="index.jsp">zum Start </a>
    </body>
</html>
```

In web.xml wird für fehlerhafte Logins noch eine Fehlerseite ergänzt. Der Eintrag dazu sieht wie folgt aus.

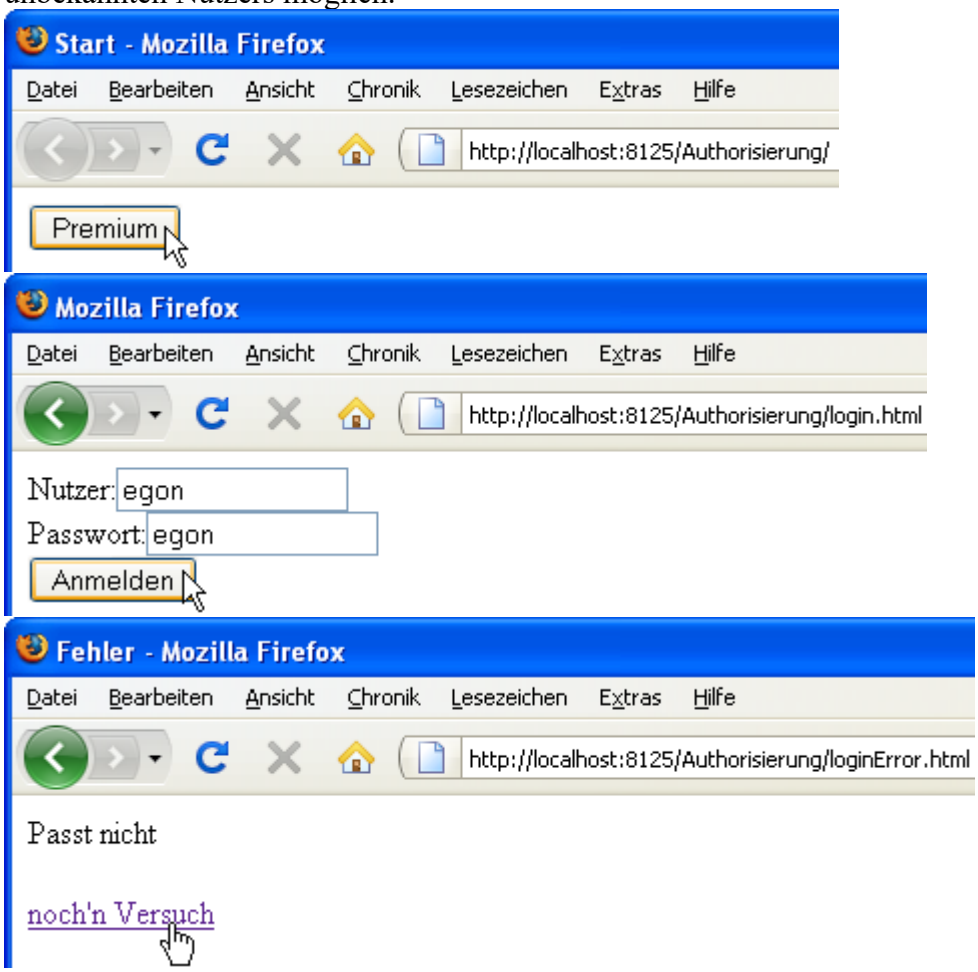
```
    <error-page>
        <error-code>403</error-code>
        <location>/fehler.jsp</location>
    </error-page>
```

Die zugehörige Seite fehler.jsp sieht wie folgt aus.

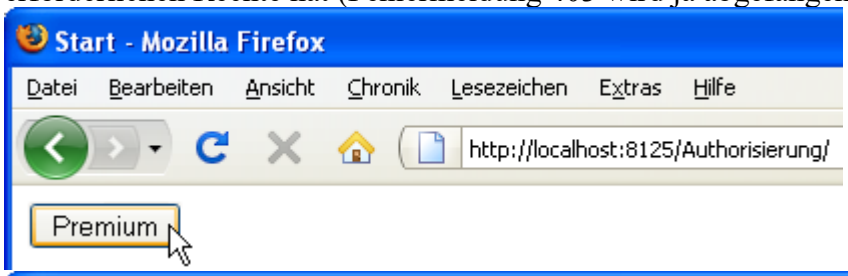
```
<%@page contentType="text/html" pageEncoding="UTF-8" session="false" isErrorPage="true"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

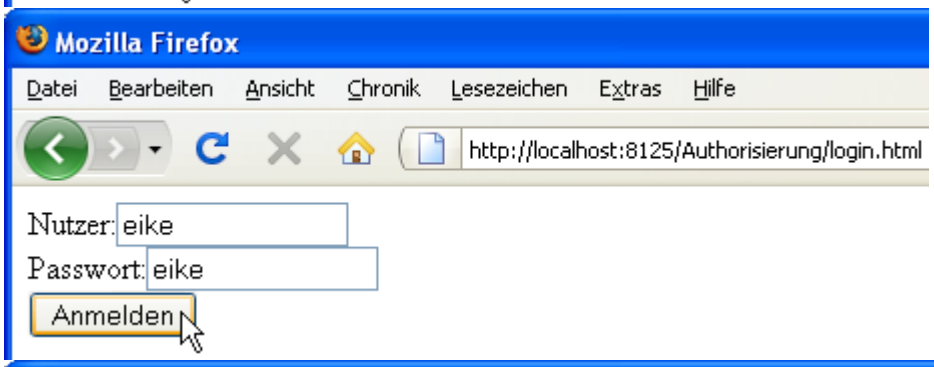
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Fehler</title>
    </head>
   <body>
       <h3>Interner Fehler</h3>
      <a href="/Authorisierung/index.jsp">zum Start </a>
      <% request.getSession().invalidate();%>
    </body>
</html>
```

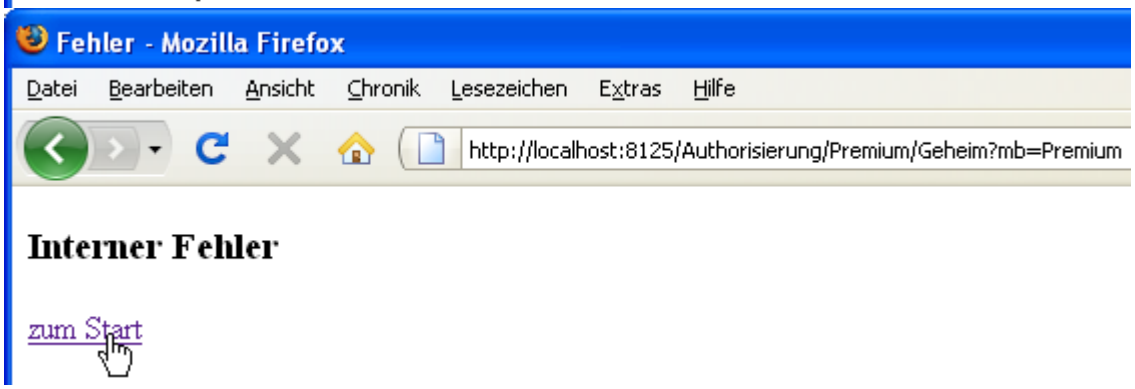
Startet man jetzt die Applikation ist folgender Ablauf für eine erfolglose Anmeldung eines unbekannten Nutzers möglich.



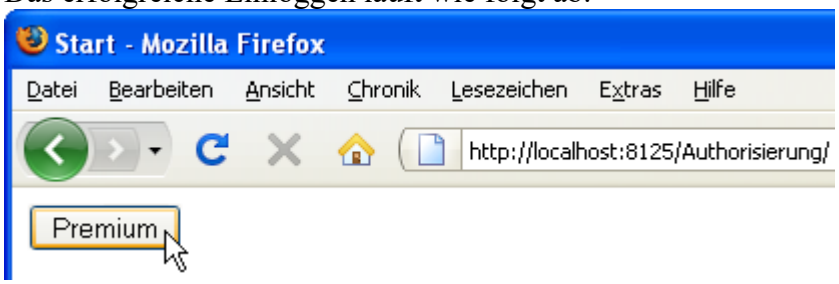
Folgender Ablauf passiert bei einem Nutzer, der dem Server zwar bekannt ist, aber nicht die erforderlichen Rechte hat (Fehlermeldung 403 wird ja abgefangen).

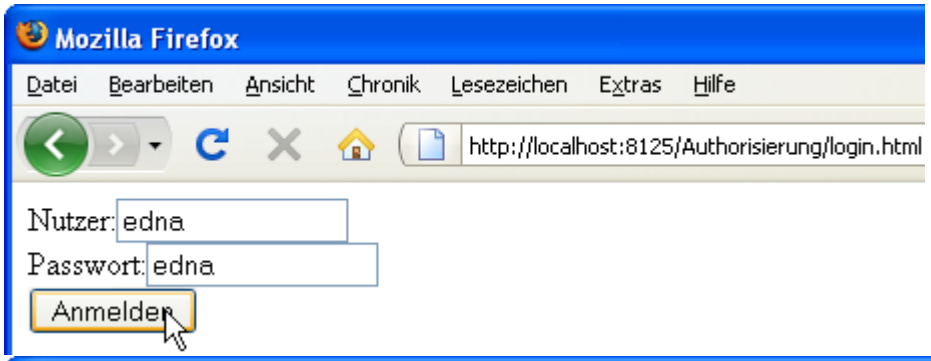


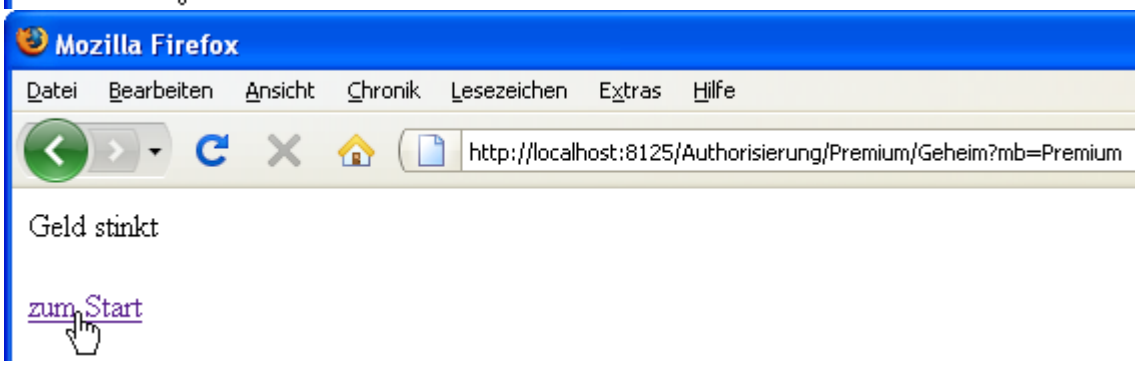


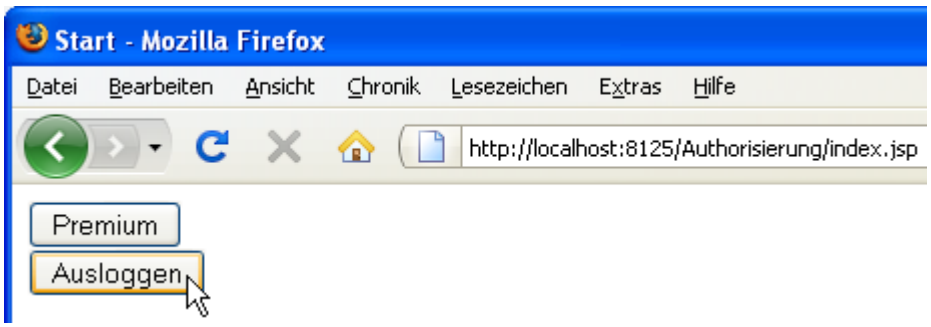


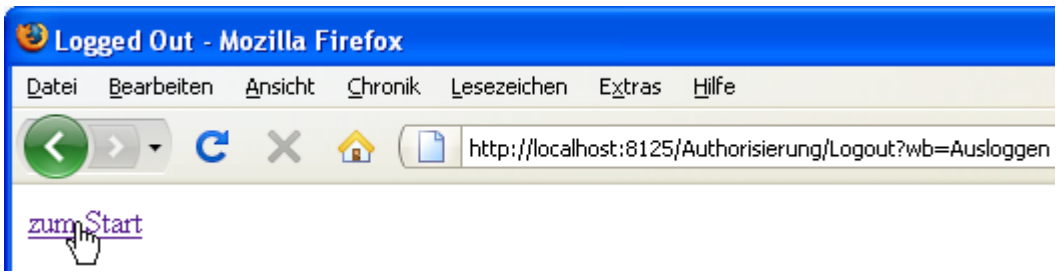
Das erfolgreiche Einloggen läuft wie folgt ab.



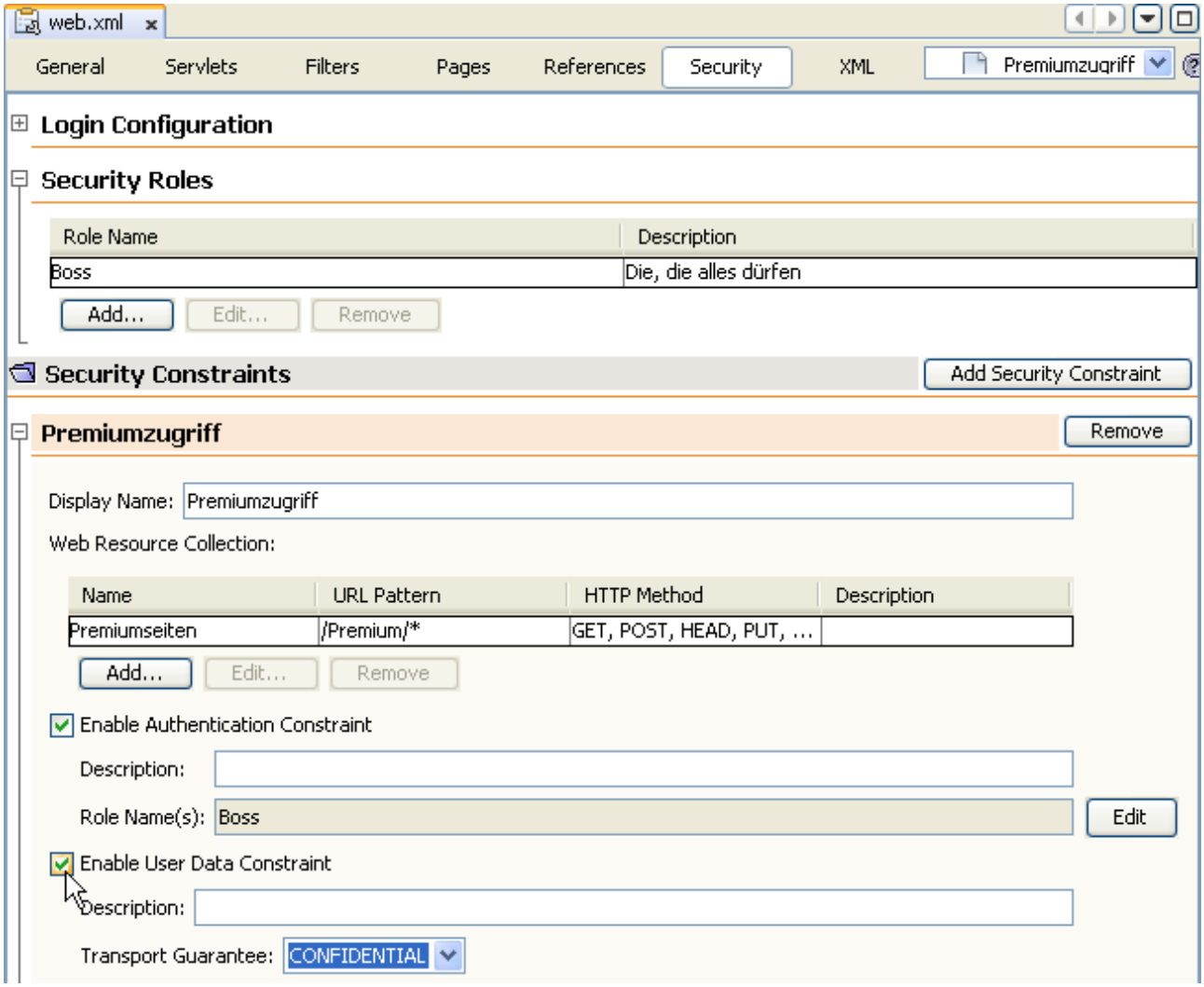




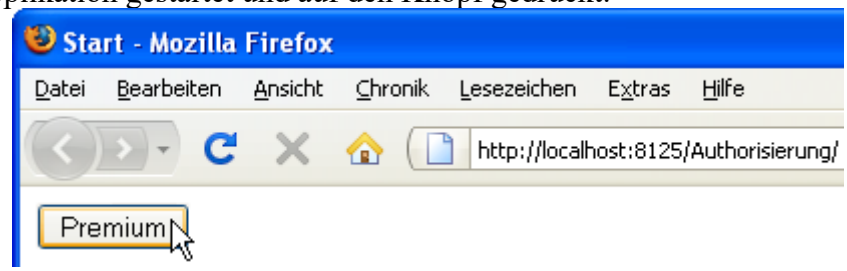




Bis jetzt wurde das Passwort unsicher übertragen. Um die sichere Übertragung zu gewährleisten, muss die so genannte „Transport Guarantee“ gesetzt werden. Dadurch wird der Client gezwungen ein Zertifikat zu nutzen. Real muss so ein Zertifikat von einer Zertifizierungsstelle erstellt werden; für lokale Testzwecke kann man eine Ausnahme einrichten, was natürlich die gesamte Konfiguration des Browsers etwas beeinflusst. Die Einstellung wird in der web.xml unter dem Reiter „Security“ unten mit einem Haken bei „Enable User Data Constraint“ als „Transport Guarantee“ eingestellt.



Jetzt wird die Applikation gestartet und auf den Knopf gedrückt.



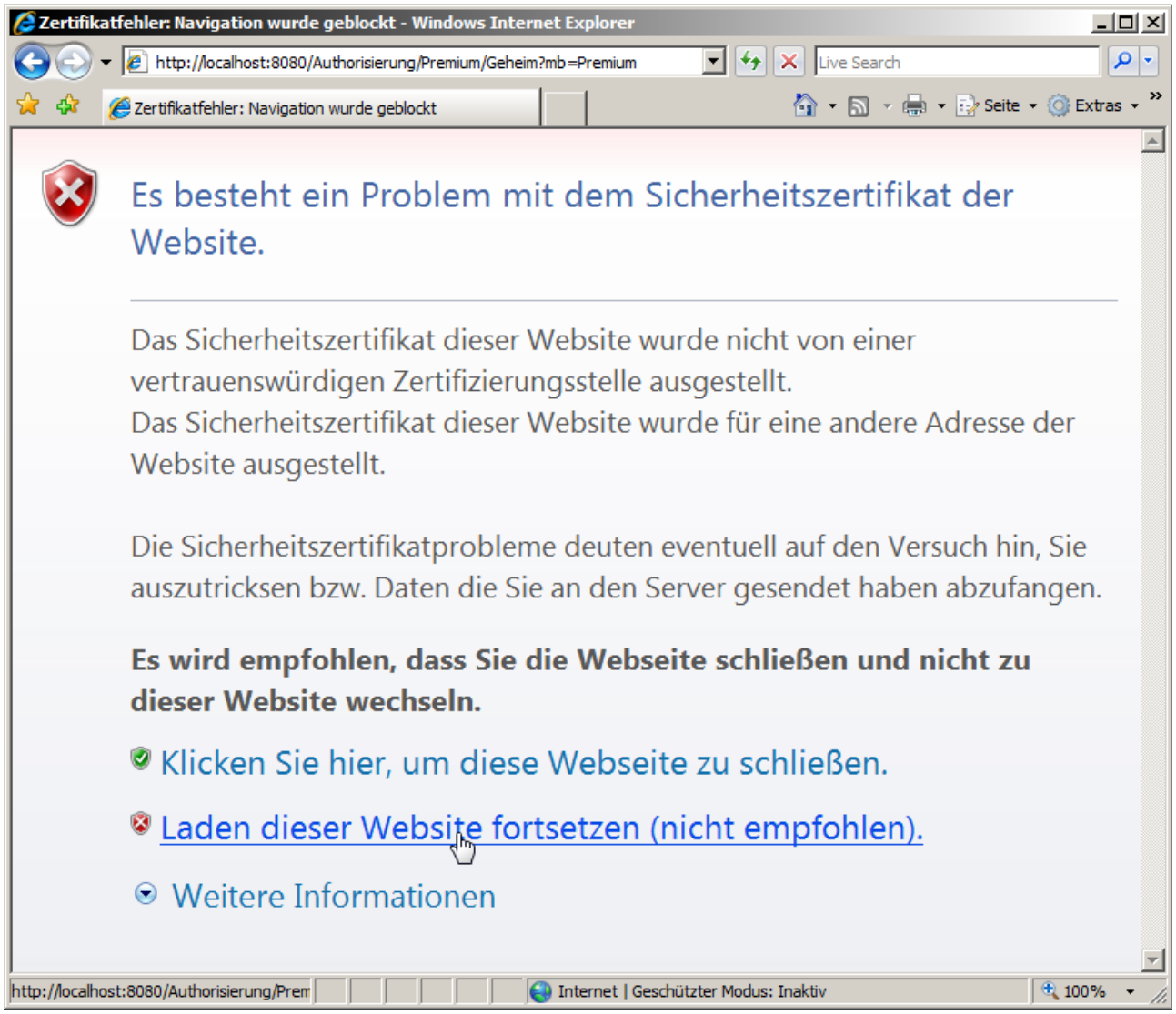
Abhängig von den Sicherheitseinstellungen, muss man im Browser die Ausführung der Applikation erlauben. Die hier notwendigen Schritte werden in dem folgenden, eingeschobenen Kapitel beschrieben.

8.7 HTTPS-Sicherheitseinstellungen in Browsern für JSP und JSF

Möchte man zur Absicherung von Applikationen das HTTPS-Protokoll nutzen, benötigt man für die

Applikation ein Zertifikat (dass man sich für den lokalen Rechner auch selber basteln könnte). Alternativ ruft man die geschützten Seiten auf, wird auf die Sicherheitsproblematik hingewiesen und kann sie behandeln. Hier werden Ansätze für den Internet-Explorer und Firefox beschrieben.

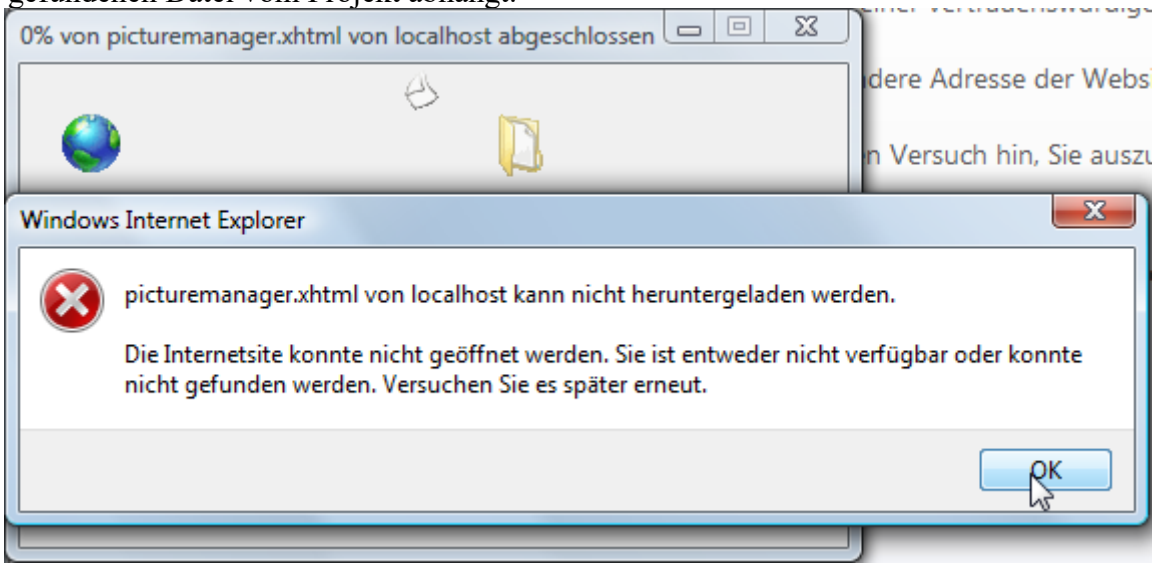
Beim Internet-Explorer erhält man folgende Meldung, bei der man die Ausnahme dann bestätigen muss. Man beachte, dass das Protokoll auf https nach der Bestätigung geändert wird. Die Ausnahme sieht im Internet Explorer wie folgt aus; durch einen Klick auf den Link kommt man trotzdem zu Anmeldebildschirm.



Die nachfolgende Warnung muss man mit „Ja“ bestätigen.



Alternativ kann der Internet-Explorer auch folgende Antwort liefern, wobei der Titel der angeblich nicht gefundenen Datei vom Projekt abhängt.



Sollte es sich um JSF-Projekt mit Facelets handeln, kann es sich um einen Konfigurationsproblem in der web.xml handeln. Für JSP-Programme ist der folgende Eintrag in Ordnung für JSF aber nicht.

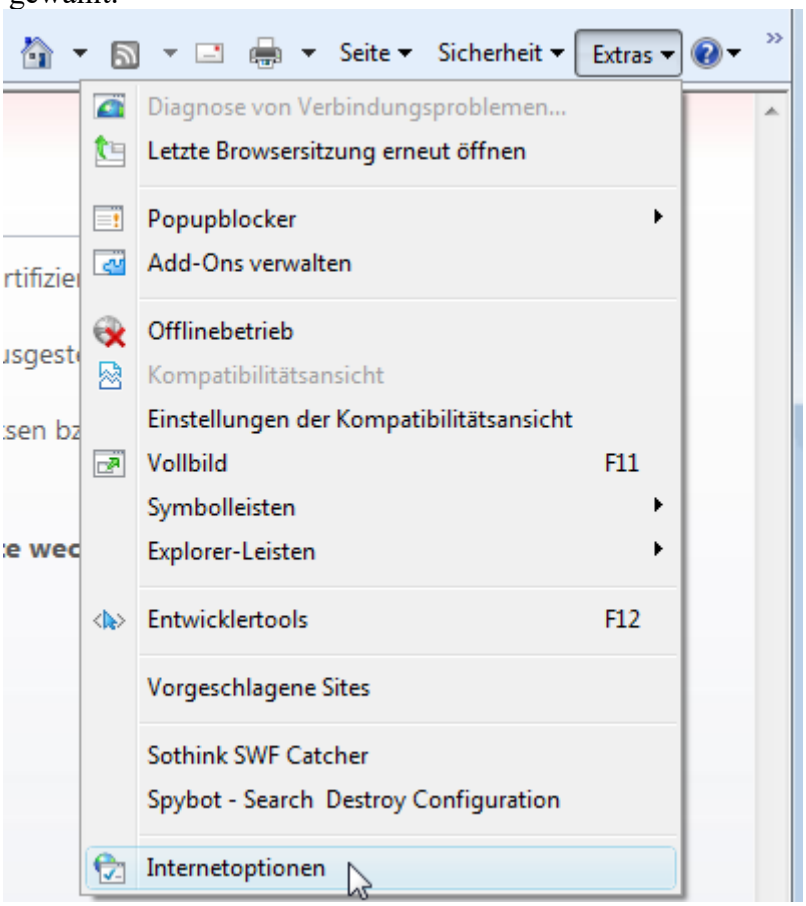
```
<login-config>
      <auth-method>FORM</auth-method>
      <form-login-config>
          <form-login-page>/login.xhtml</form-login-page>
          <form-error-page>/loginError.xhtml</form-error-page>
      </form-login-config>
  </login-config>
```

Die Verweise auf die Seiten müssen bei den Standardeinstellungen berücksichtigen, dass sie aus Sicht des Projekts in den Ordner faces geschoben sind. Der weiterhin ergänzte realm-name-Eintrag deutet nur weitere Möglichkeiten zur systematischeren Nutzerverwaltung an.

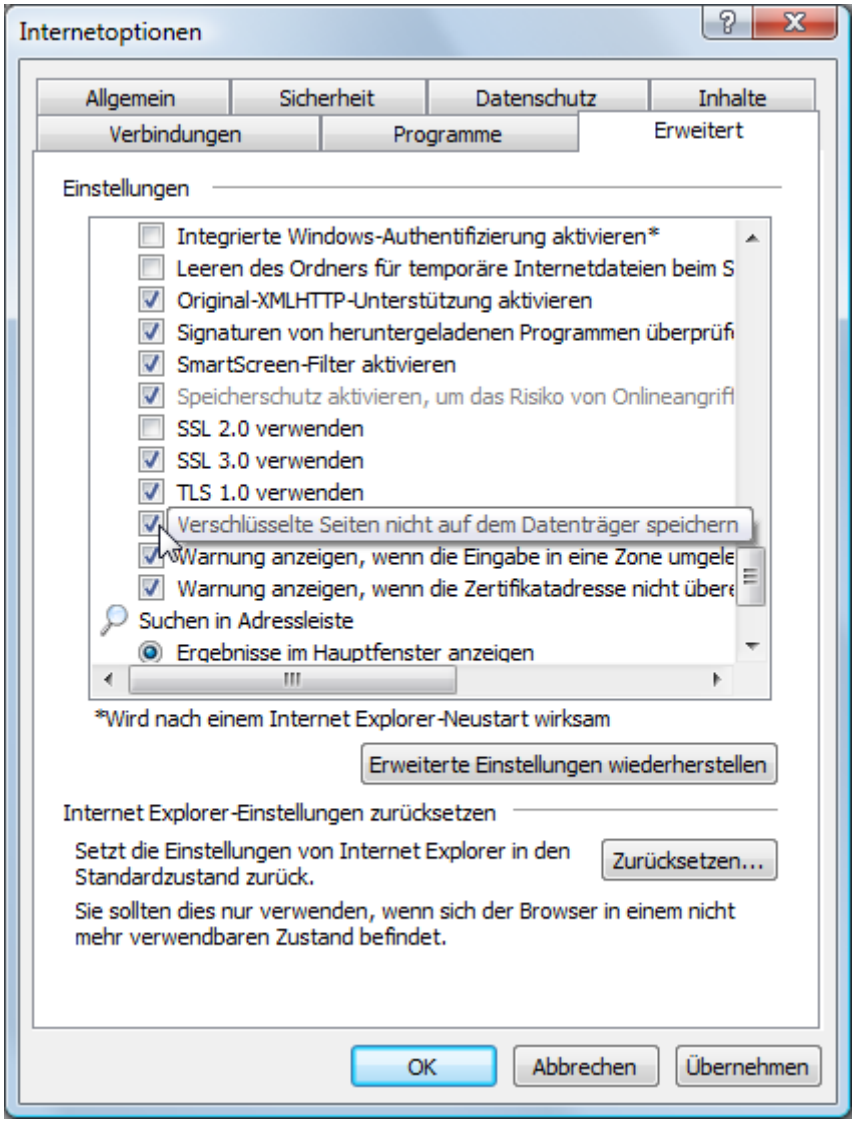
```
<login-config>
      <auth-method>FORM</auth-method>
      <realm-name>file</realm-name>
```

```
        <form-login-config>
            <form-login-page>/faces/login.xhtml</form-login-page>
            <form-error-page>/faces/accessDenied.xhtml</form-error-page>
        </form-login-config>
    </login-config>
```

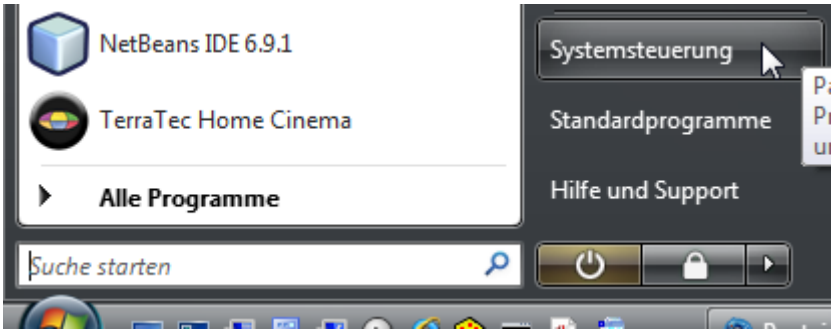
Falls es sich nicht um das oben beschriebene Problem handelt, müssen die Sicherheitseinstellungen des Internet Explorers angepasst werden. Dazu wird rechts oben der Punkt „Extras“ und dann „Internetoptionen“ gewählt.



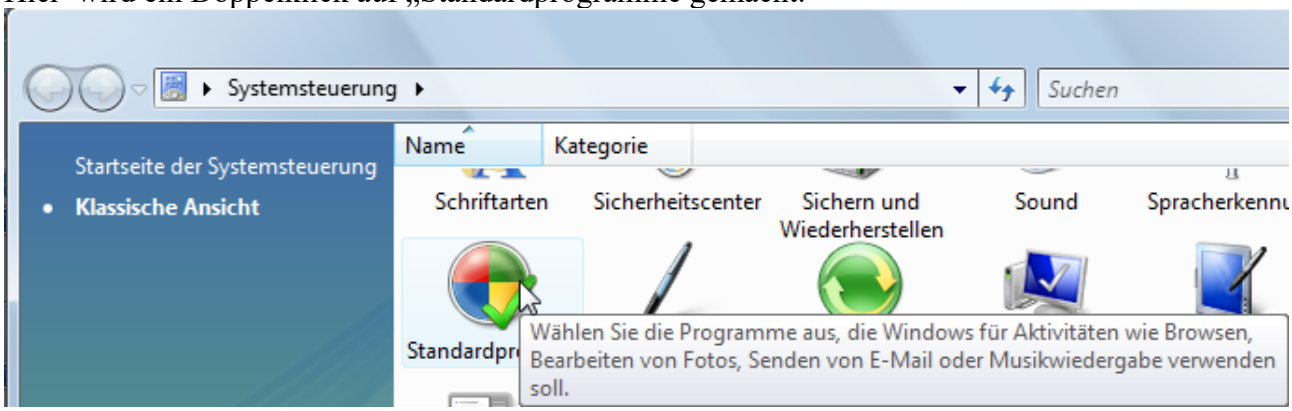
Hier wird der Reiter „Extras“ gewählt und dann im großen Fenster der Punkt „Sicherheit“ zu scrollen. Hier befindet sich nach einigem Suchen der Punkt „Verschlüsselte Seiten nicht auf dem Datenträger speichern“. Hier muss ein Haken gesetzt und über „OK“ bestätigt werden.



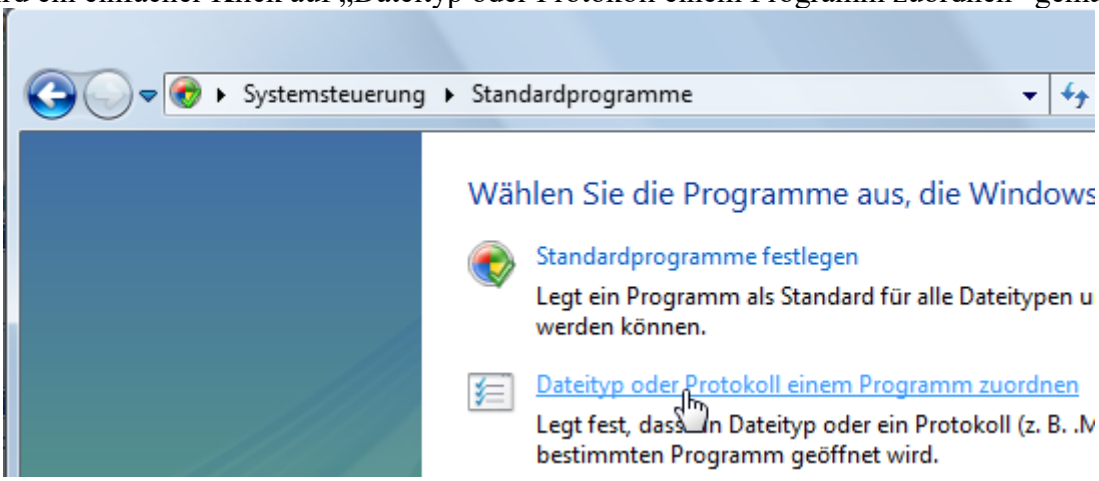
Bleibt der Fehler bestehen, kann eine der vielen Fehlerquellen (bis zum Virus!) sein, dass Programmendungen mit „falschen“ Programmen zum Öffnen verknüpft sind. Im folgenden Beispiel hat der Nutzer während anderer Entwicklungen festgelegt, dass XML-Dateien nicht mit dem Internet-Explorer, sondern mit „Word Pad“ geöffnet werden sollen. Diese Einstellung muss rückgängig gemacht werden. Unter Windows Vista kann man hier z. B. folgenden Weg wählen. Beim Startknopf wird „Systemsteuerung“ gewählt.



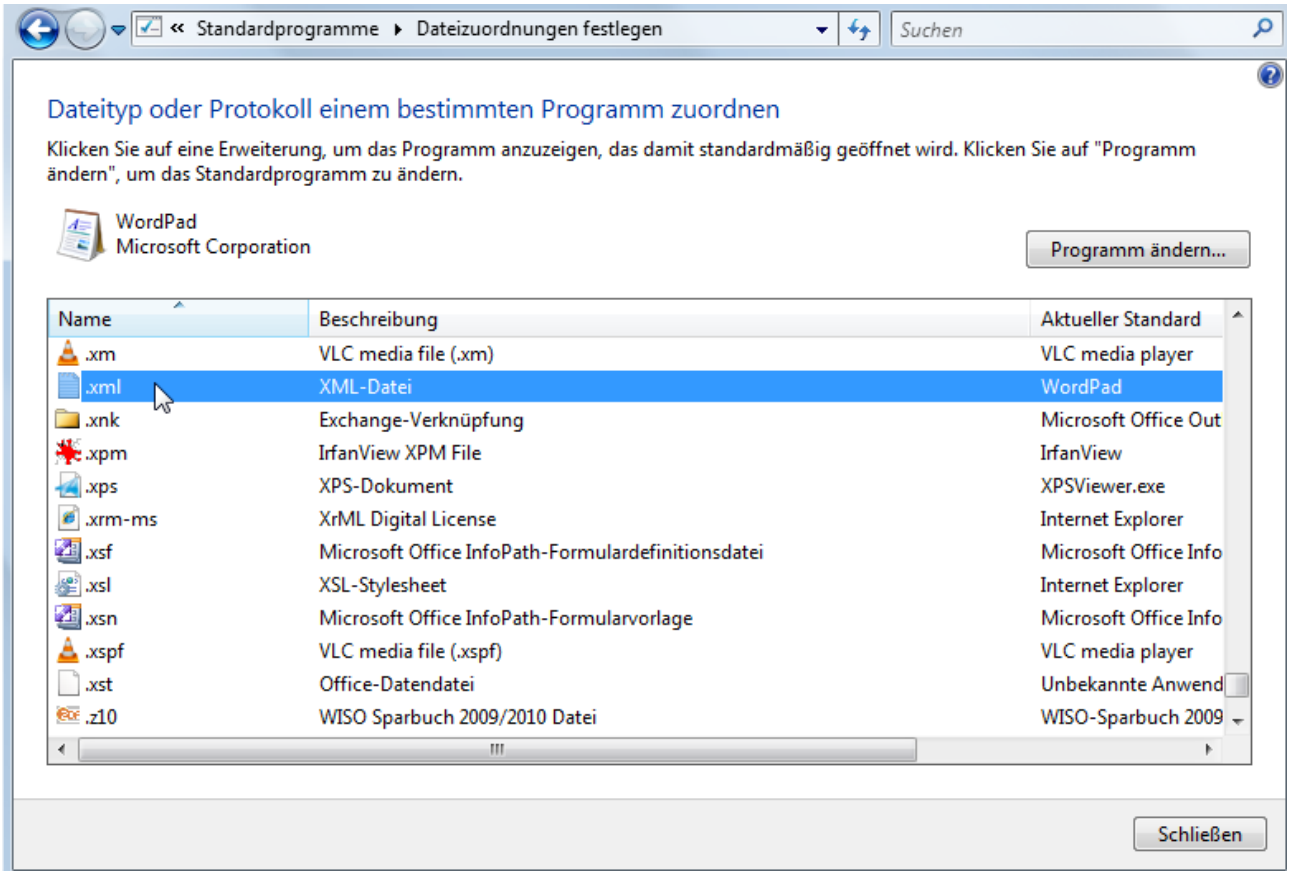
Hier wird ein Doppelklick auf „Standardprogramme gemacht.



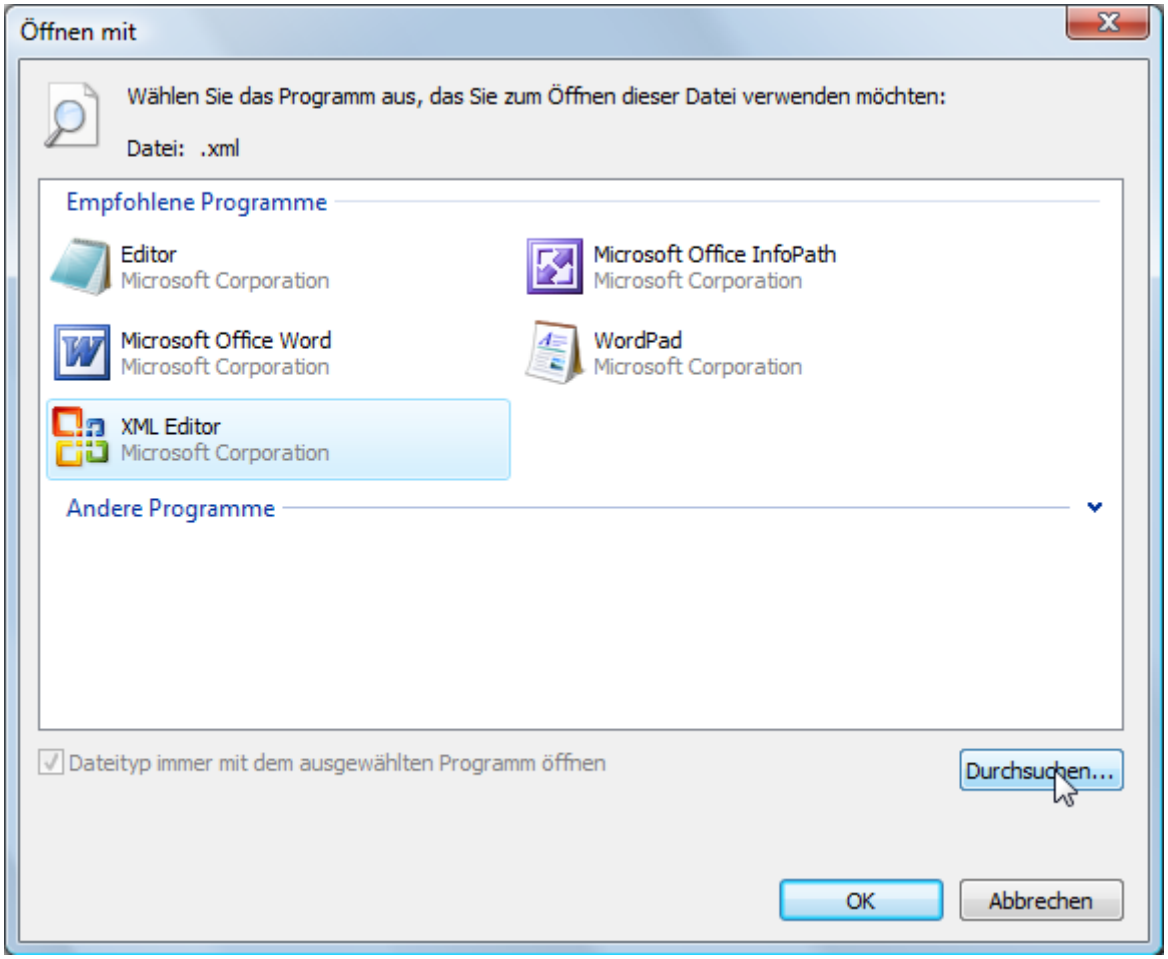
Nun wird ein einfacher Klick auf „Dateityp oder Protokoll einem Programm zuordnen“ gemacht.



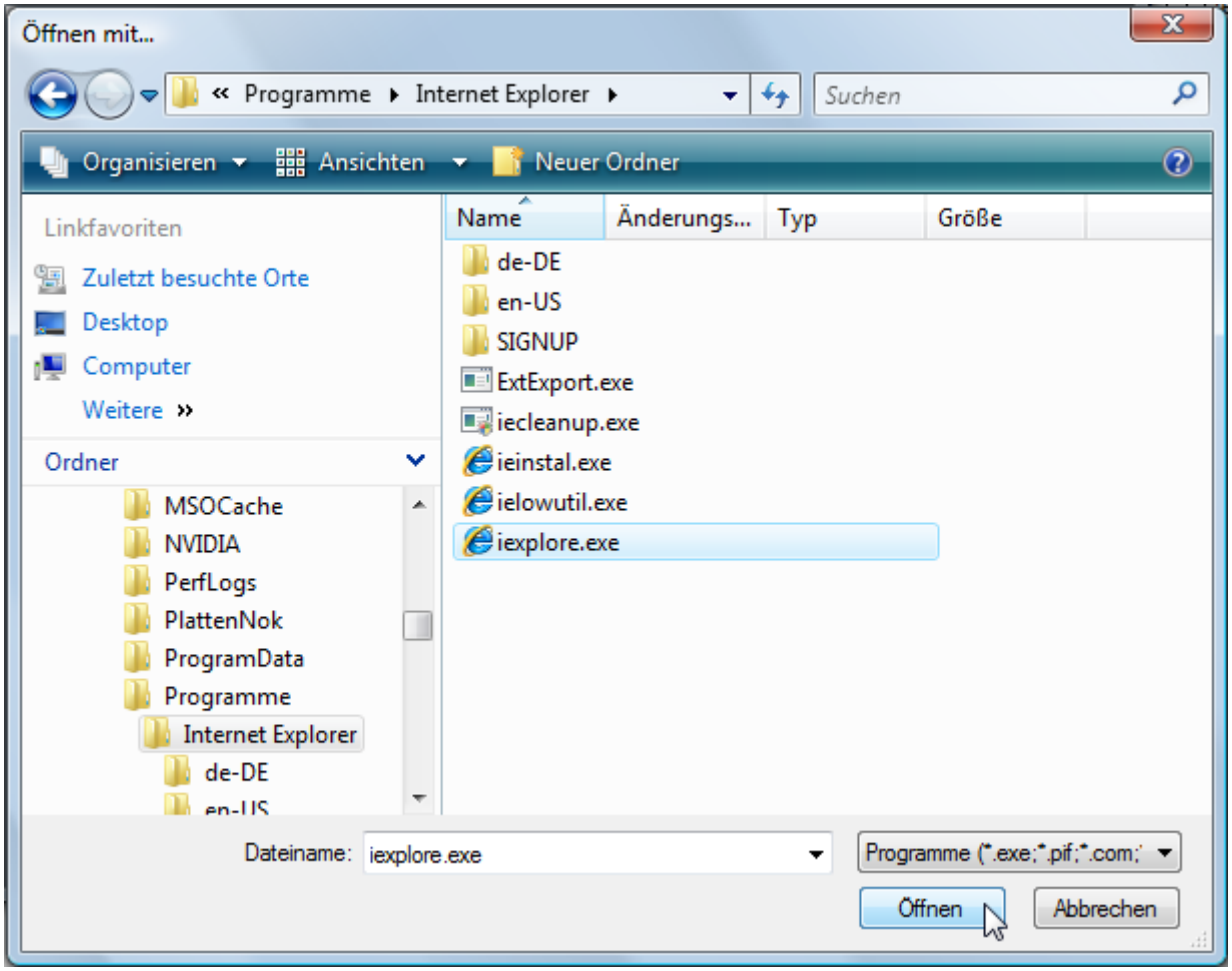
Im mittleren größeren Fenster bekommt man links die Dateiendung und rechts das zum Öffnen benutzte Programm angezeigt. Hier wird jetzt die Endung „.xml“ gesucht.



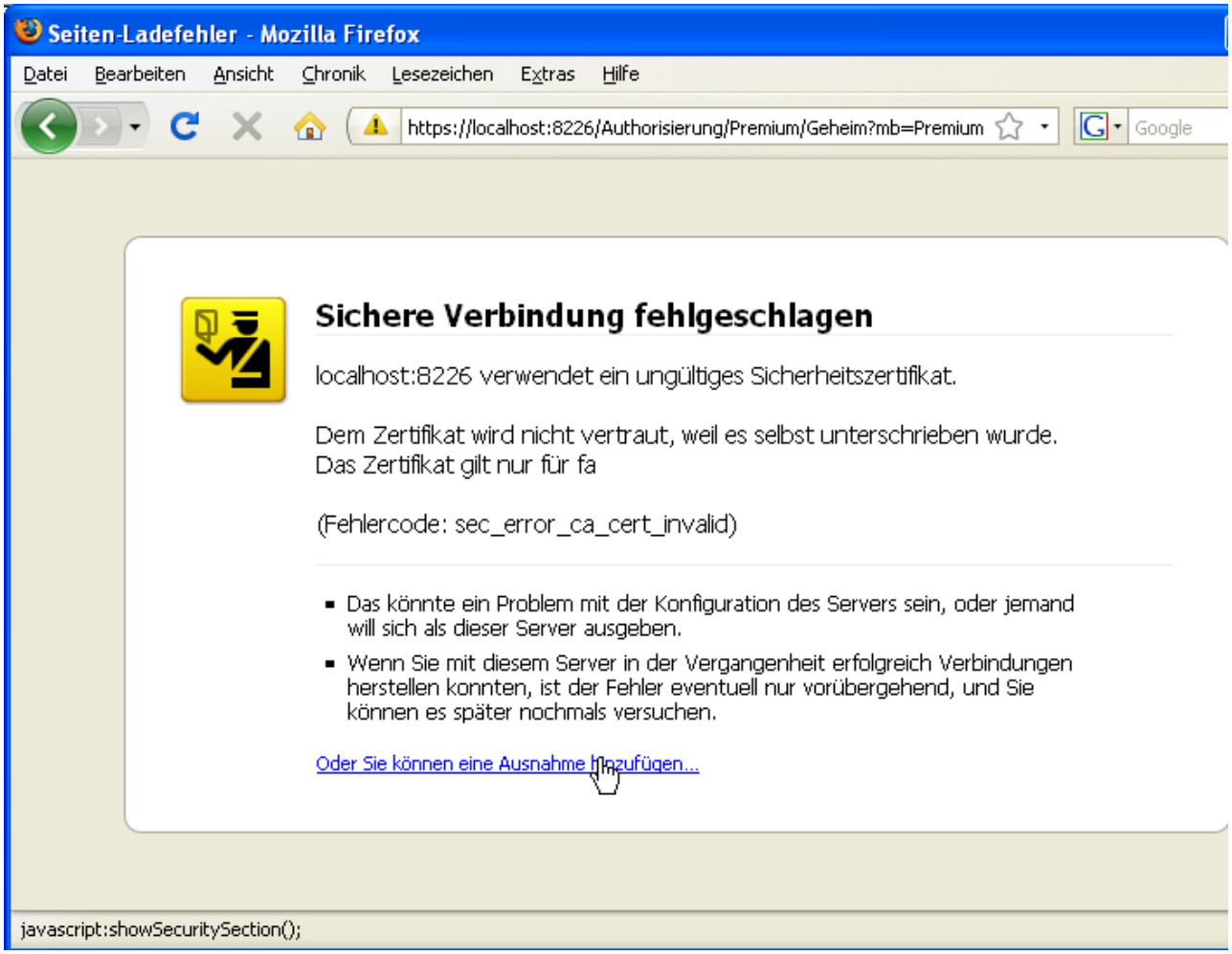
Nun klickt man auf „Programm ändern“ und stellt den Internet Explorer als Standard-Programm ein. Im Beispiel wird der Internet Explorer nicht direkt als Möglichkeit angegeben, so dass er über „Durchsuchen“ gesucht werden muss.



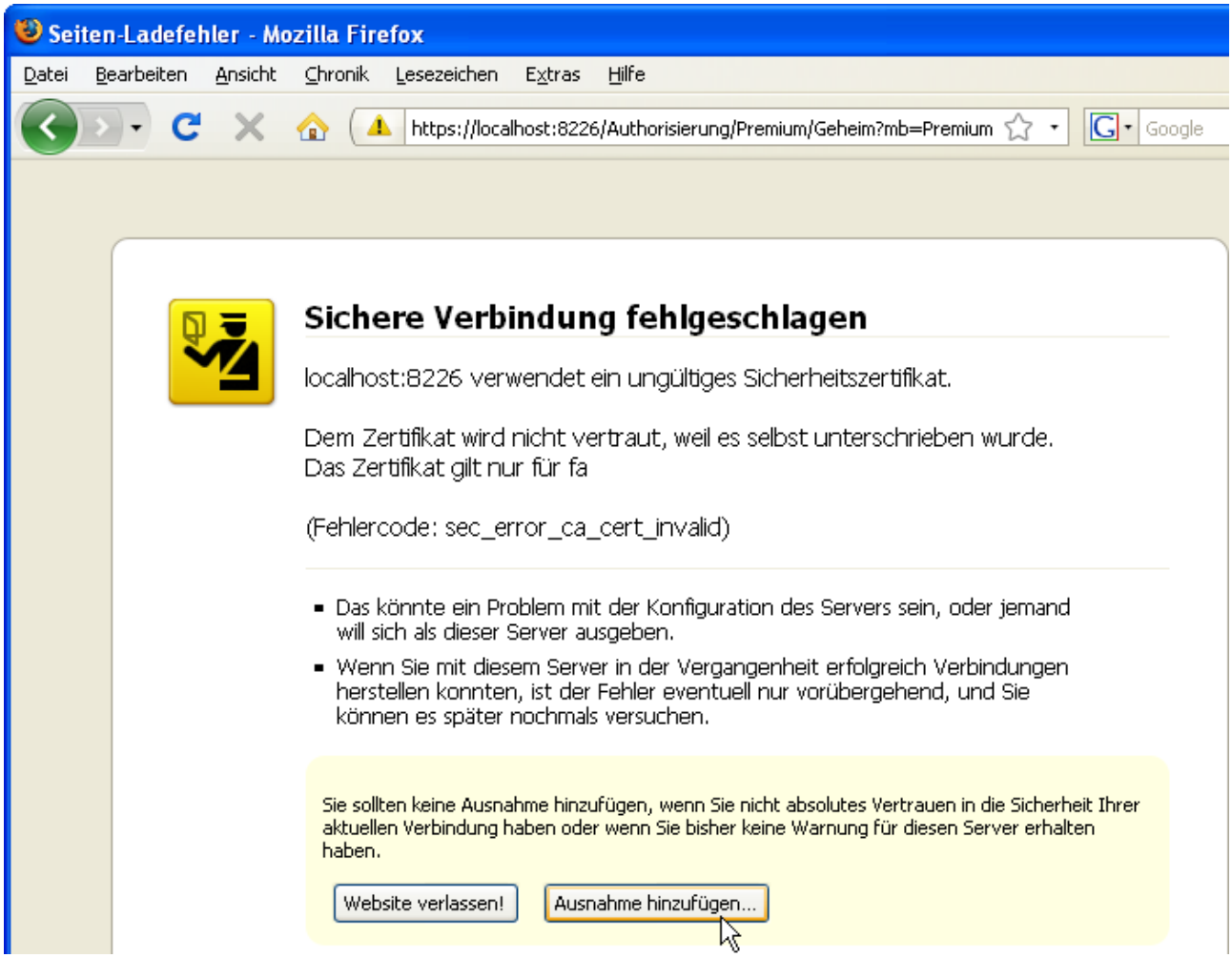
Nun muss der Internet Explorer gesucht und das ausführbare Programm ausgewählt werden. Danach werden alle Fenster über „OK“ bzw. „Schließen“ geschlossen.



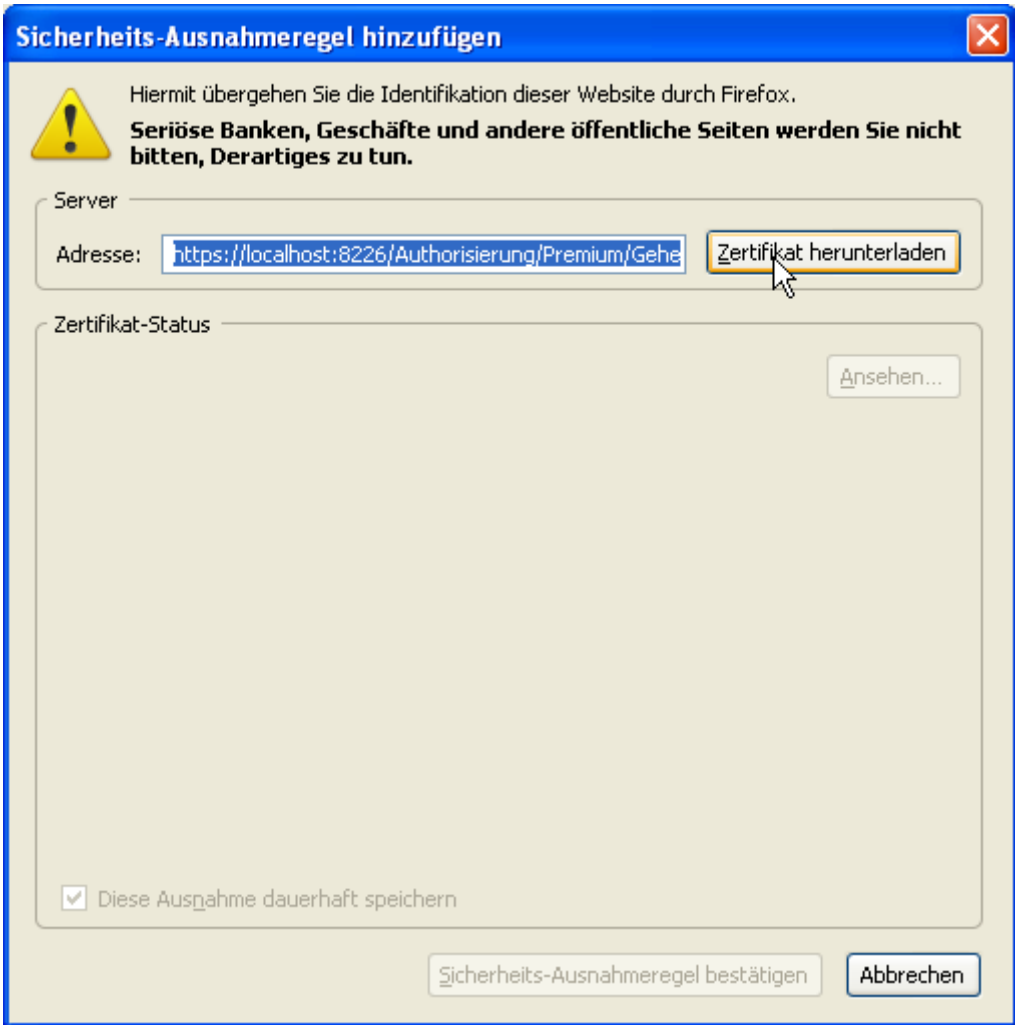
Beim Firefox hängt es von den Einstellungen ab, was passiert. Eine Variante ist die folgende Meldung, von der ausgehend eine Ausnahme eingerichtet werden kann. Eine weitere Variante wird nachfolgend vorgestellt.



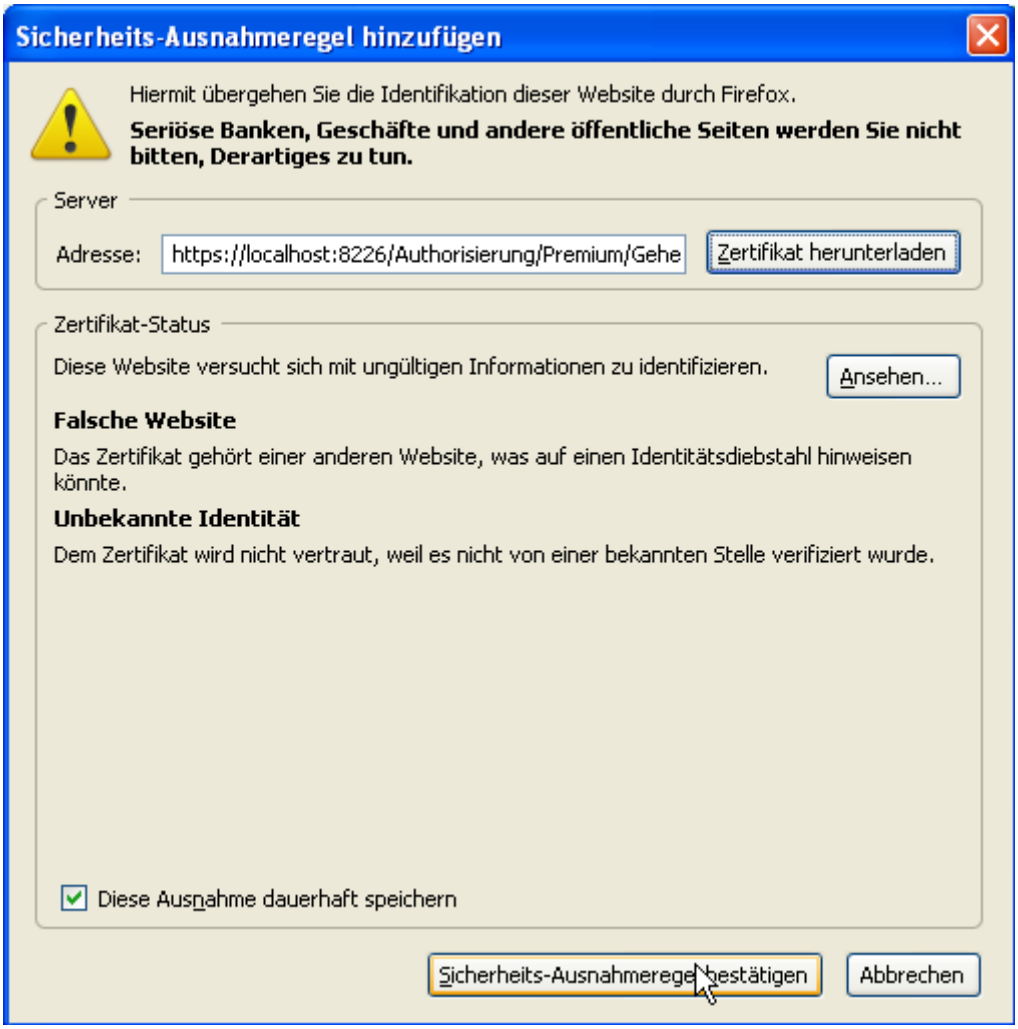
Man kann dann die Ausnahme hinzufügen.



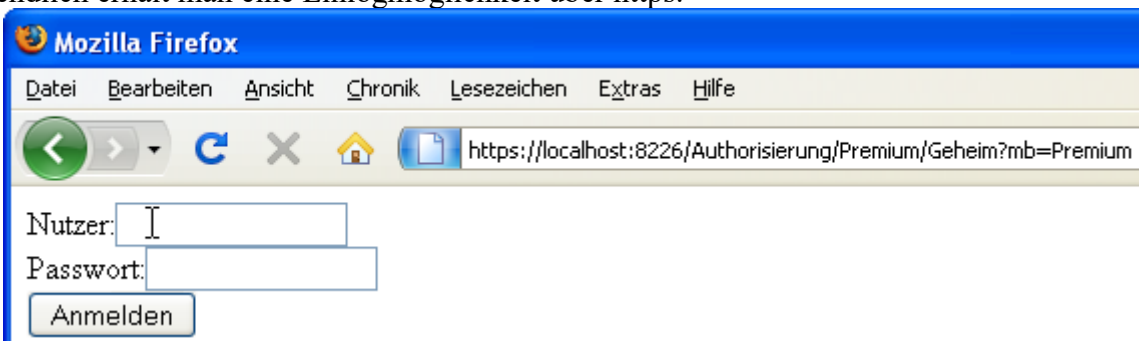
Jetzt muss ein temporäres (Test-)Zertifikat vom eigenen Rechner lokal „heruntergeladen“ werden.



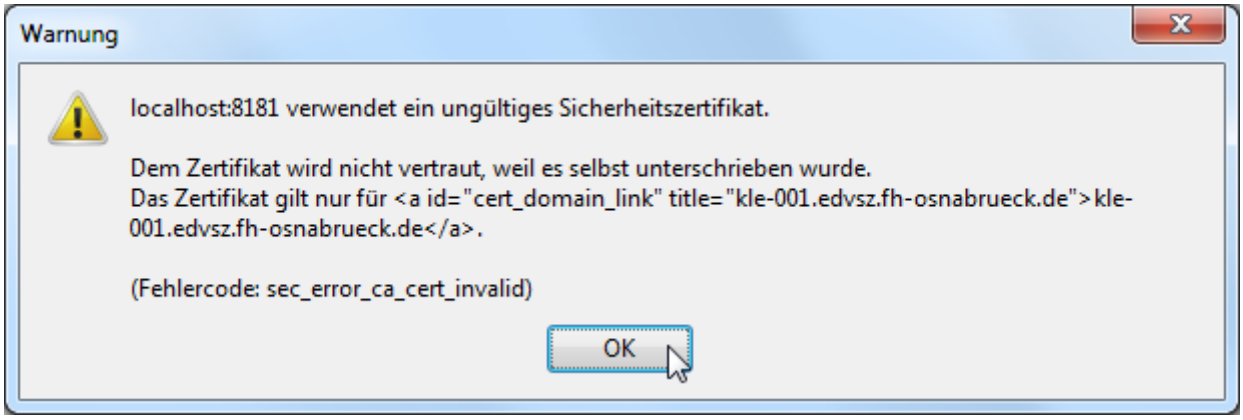
Danach kann die Ausnahme endgültig bestätigt werden.



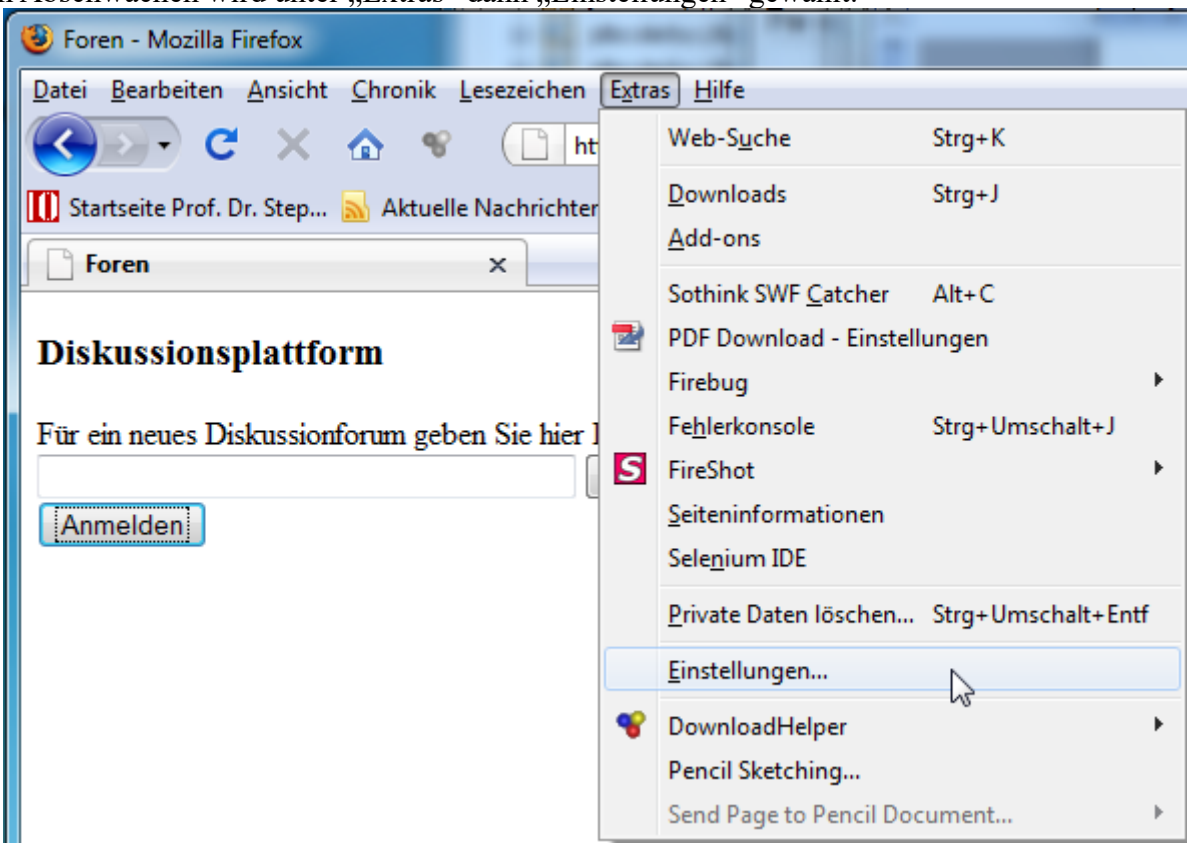
Letztendlich erhält man eine Einlogmöglichkeit über https.



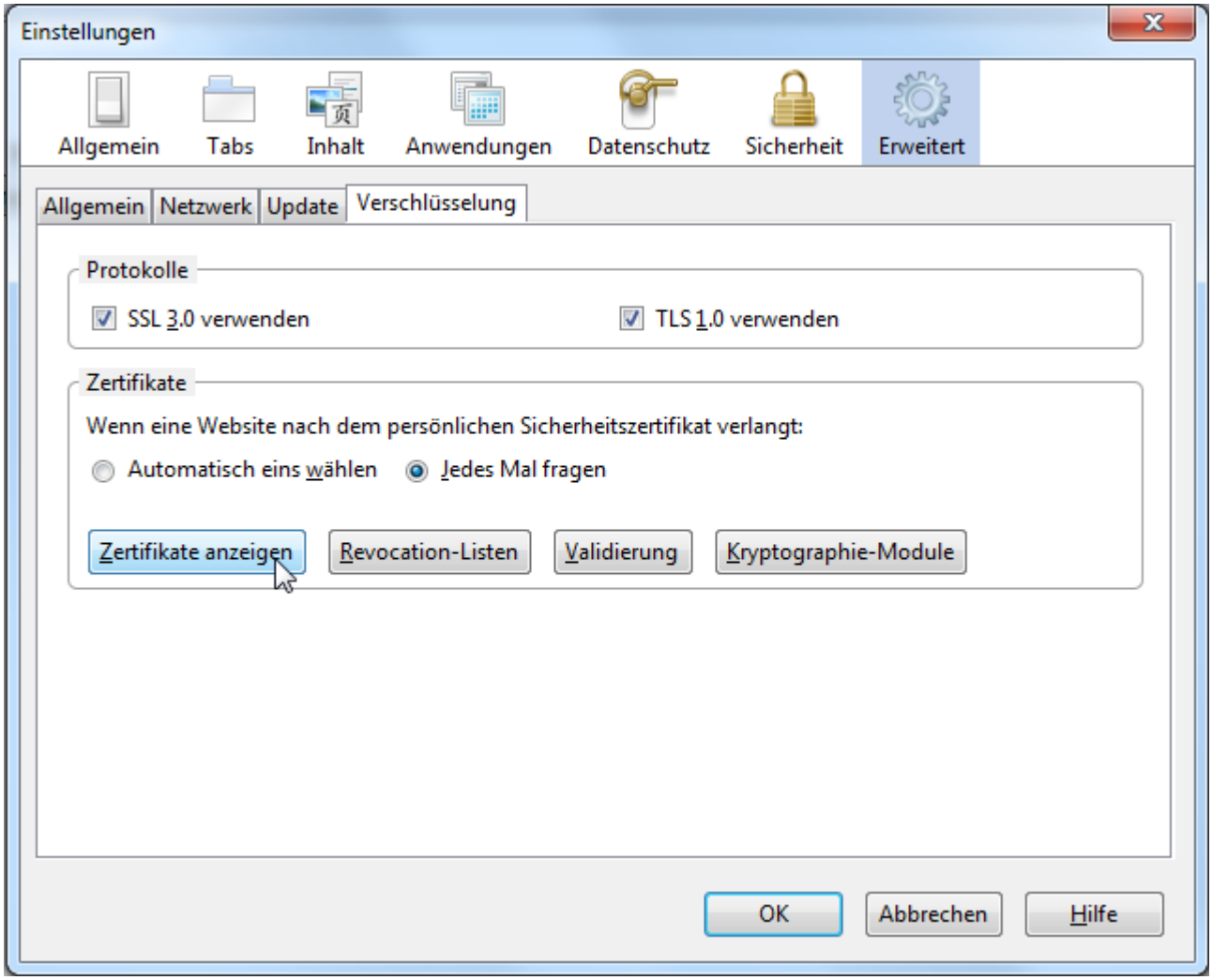
Falls Firefox eine Meldung der folgenden Form zeigt, müssen leider die Sicherheitseinstellungen für den localhost abgeschwächt werden, damit der Nutzerzugang getestet werden kann.



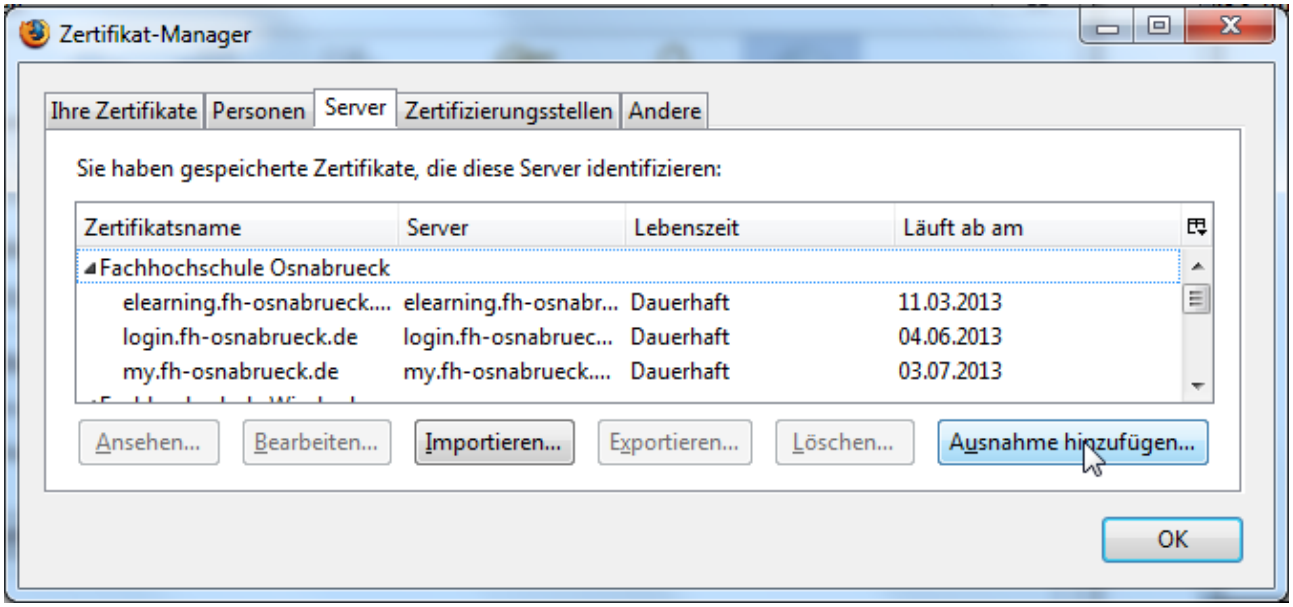
Zum Abschwächen wird unter „Extras“ dann „Einstellungen“ gewählt.



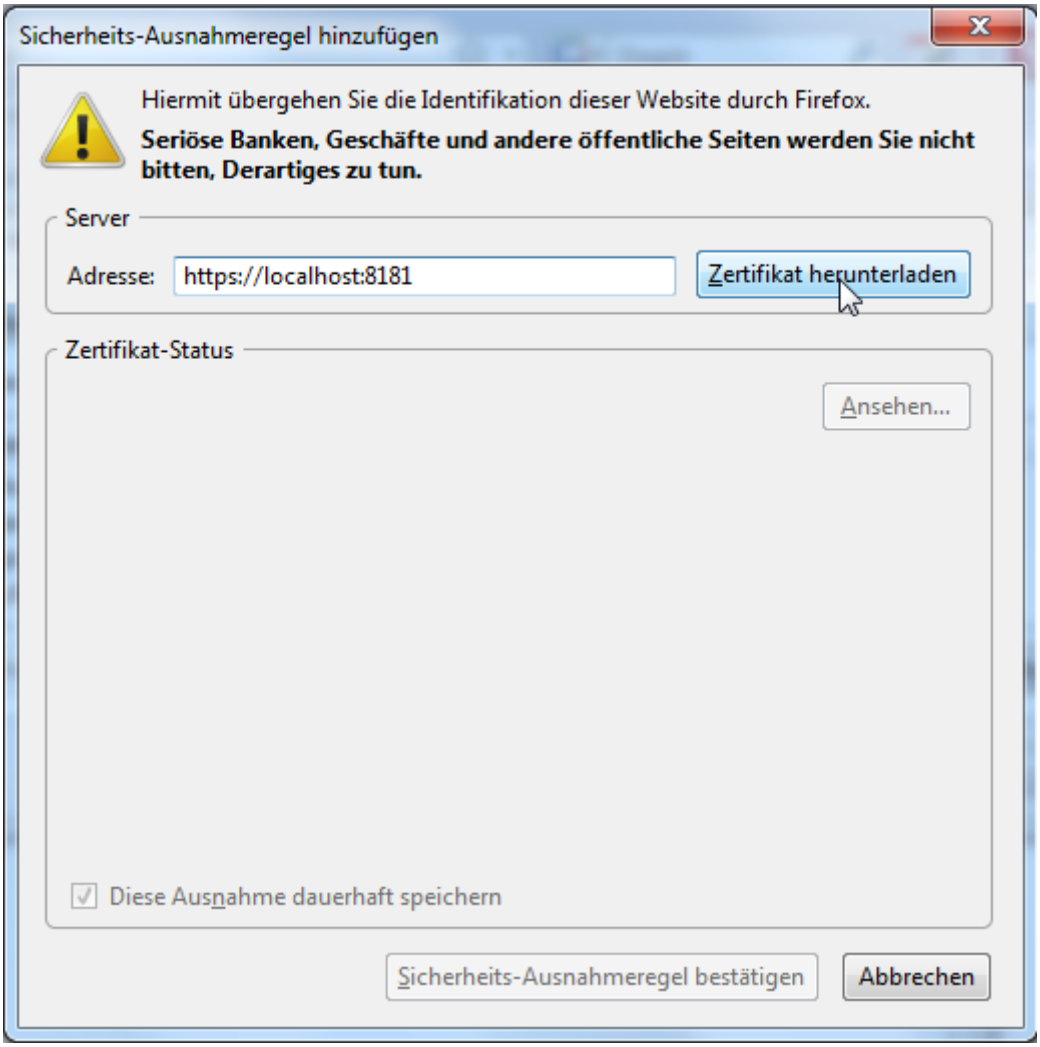
Hier wird oben der Reiter „Verschlüsselung“ gewählt und links unten auf „Zertifikate anzeigen“ gedrückt.



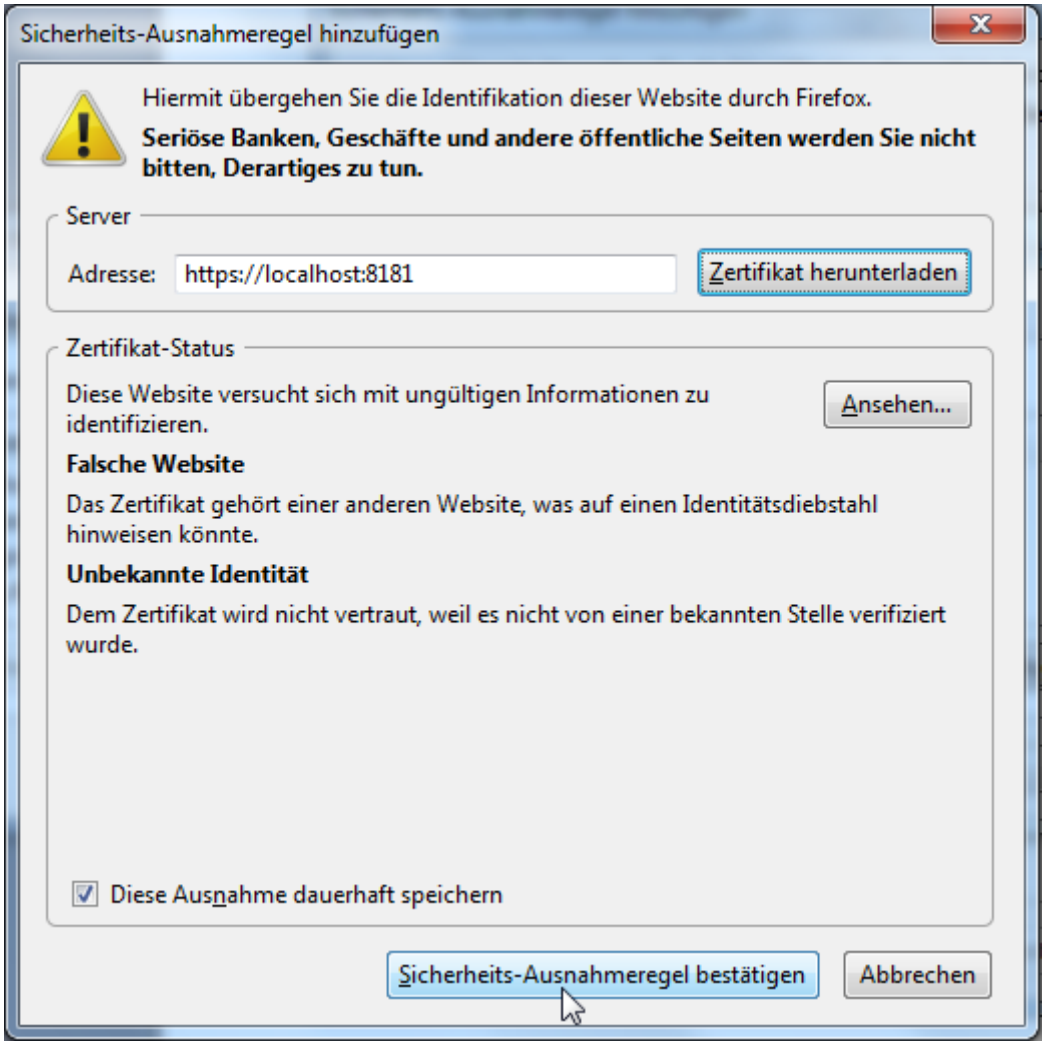
Hier wird oben der Reiter „Server gewählt“ und rechts-unten auf „Ausnahme hinzufügen…“ geklickt.



Nun wird unter Adresse, der localhost zusammen mit dem Port eingetragen, über den HTTPS läuft; dies ist üblicherweise 8181 (oder https://localhost:23475), wenn HTTP 8080 (oder http://localhost:23474) nutzt. Das Addieren von 101 (oder 1) ermöglicht beim GlassFish häufiger die Berechnung des HTTPS-Ports ausgehend vom HTTP-Port. Es wird dann auf „Zertifikat herunterladen“ geklickt.

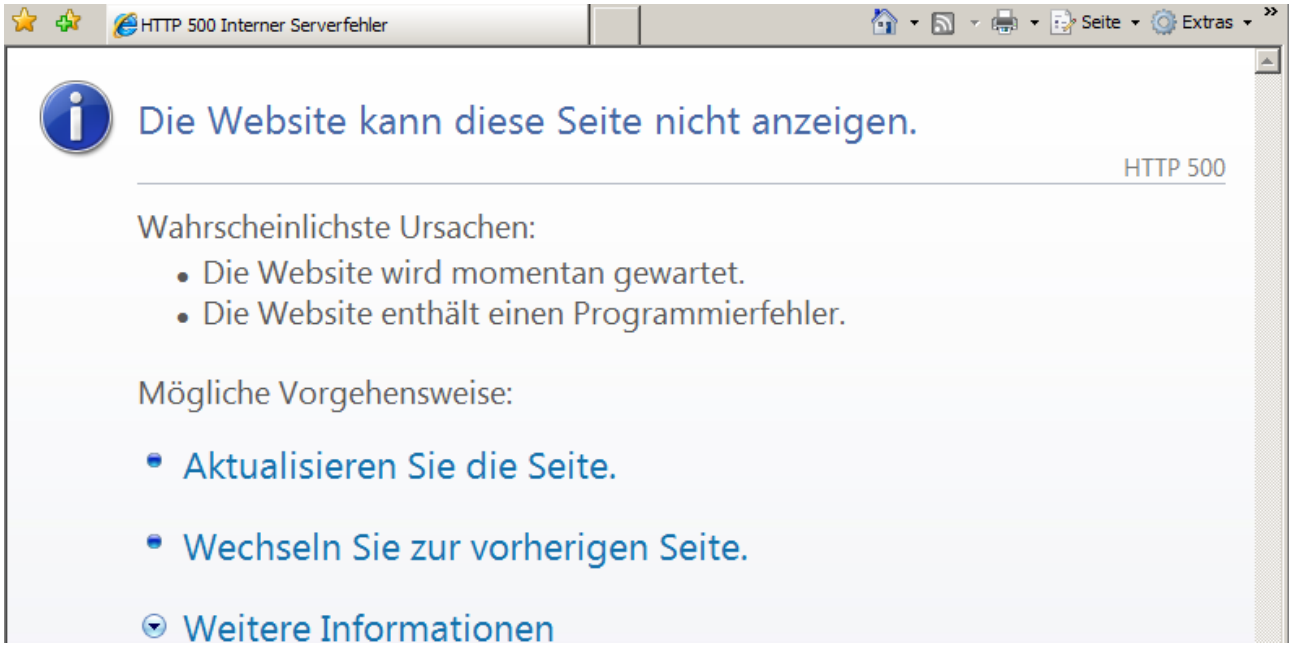


Man erhält einige Informationen und muss die Ausnahme unten durch einen Klick auf „Sicherheits-Ausnahmeregel bestätigen“ bestätigen und auf der folgenden Seite mit „OK“ beenden.

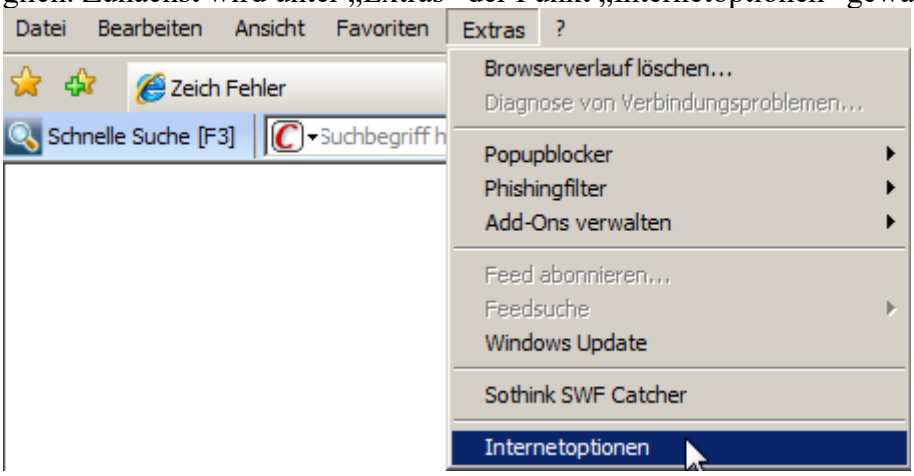


8.8 Probleme mit Fehlerseiten

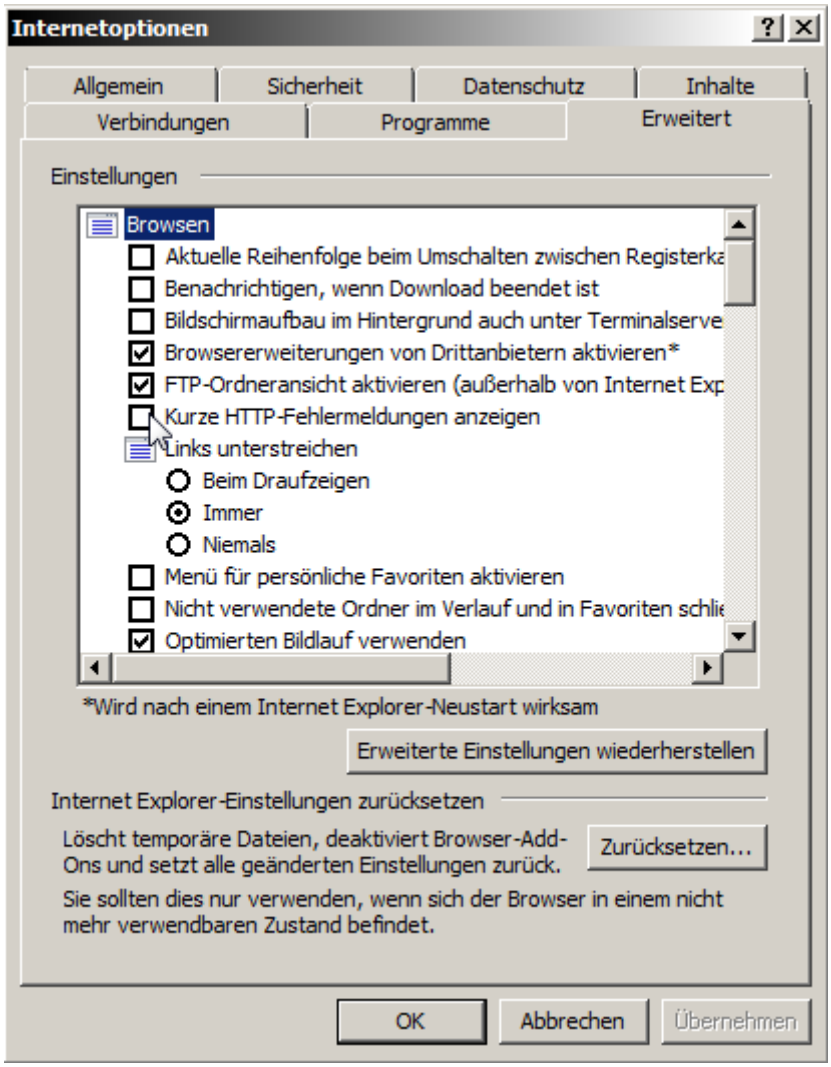
Auch bei völlig korrekter Nutzung des Fehlerseitenansatzes von JSP kann es beim Internet Explorer passieren, dass die Fehlerseite nicht angezeigt wird und man eine Seite der folgenden Form erhält.



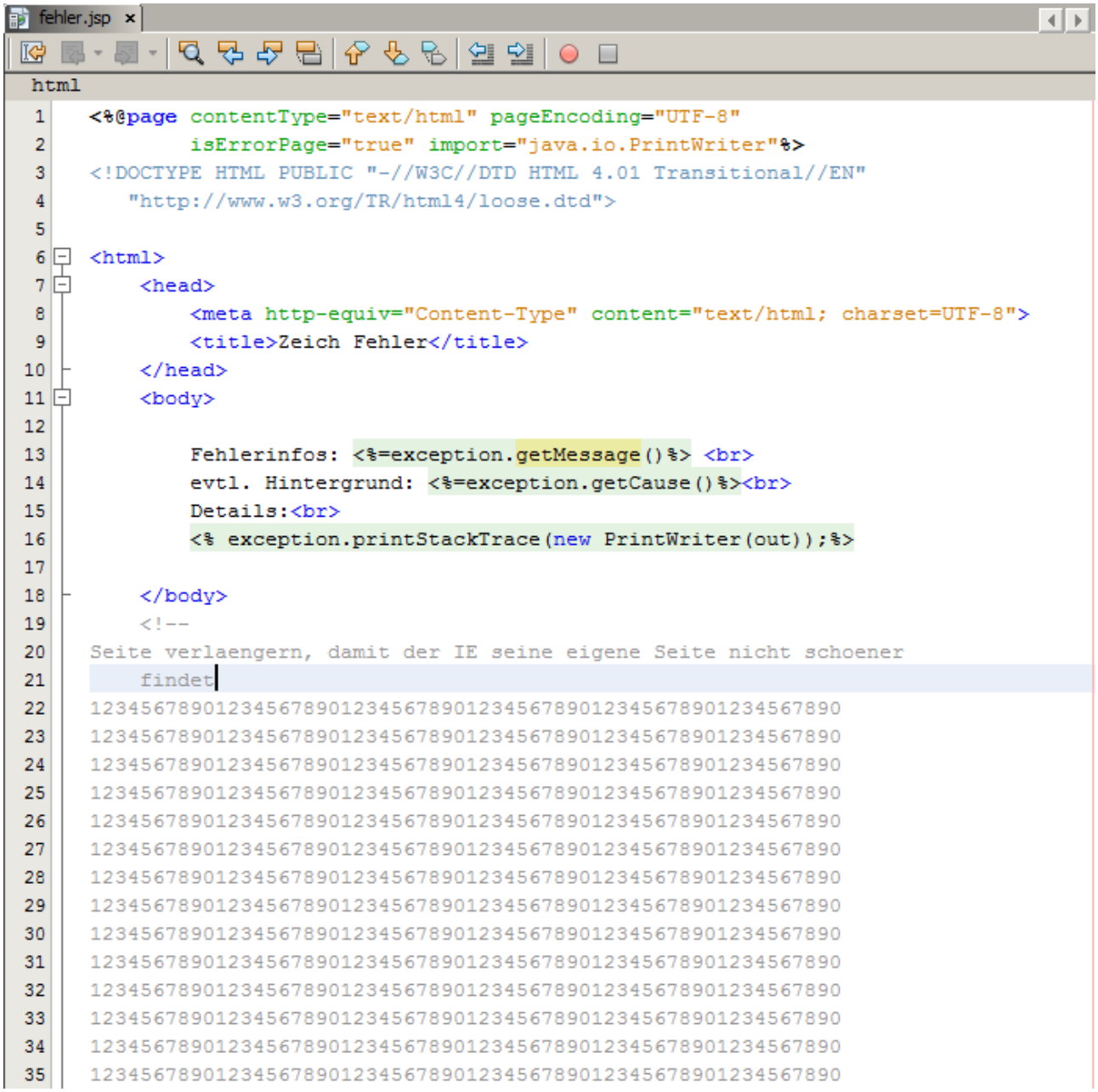
Der Internet-Explorer hat eine sogenannte fehlerfreundliche Darstellung, die beim Erhalt eines Fehlercodes und einer Seite mit weniger als 512 Bytes genutzt wird.
Zum Testen kann man auch im Internet-Explorer die fehlerfreundliche Darstellung von Problemen abschalten. Hierzu müssen die Darstellungsoptionen geändert werden. Dies ist im Internet Explorer 7 wie folgt möglich. Zunächst wird unter „Extras“ der Punkt „Internetoptionen“ gewählt.



Dann wird der Reiter „Erweitert“ gewählt und die Markierung bei „Kurze HTTP-Fehlermeldungen anzeigen“ gelöscht.



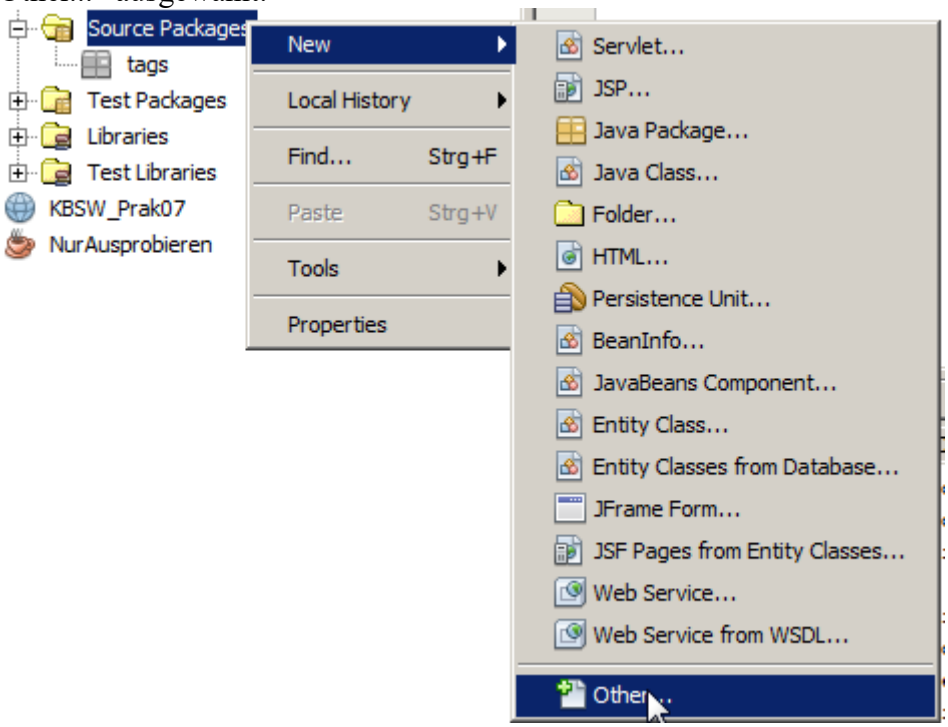
Alternativ muss man die „fehlerfreundliche Darstellung“ des Internet Explorers umgehen. Dazu muss die generierte Seite mindestens 512 Byte enthalten, was man durch einen HTML-Kommentar erreichen kann.



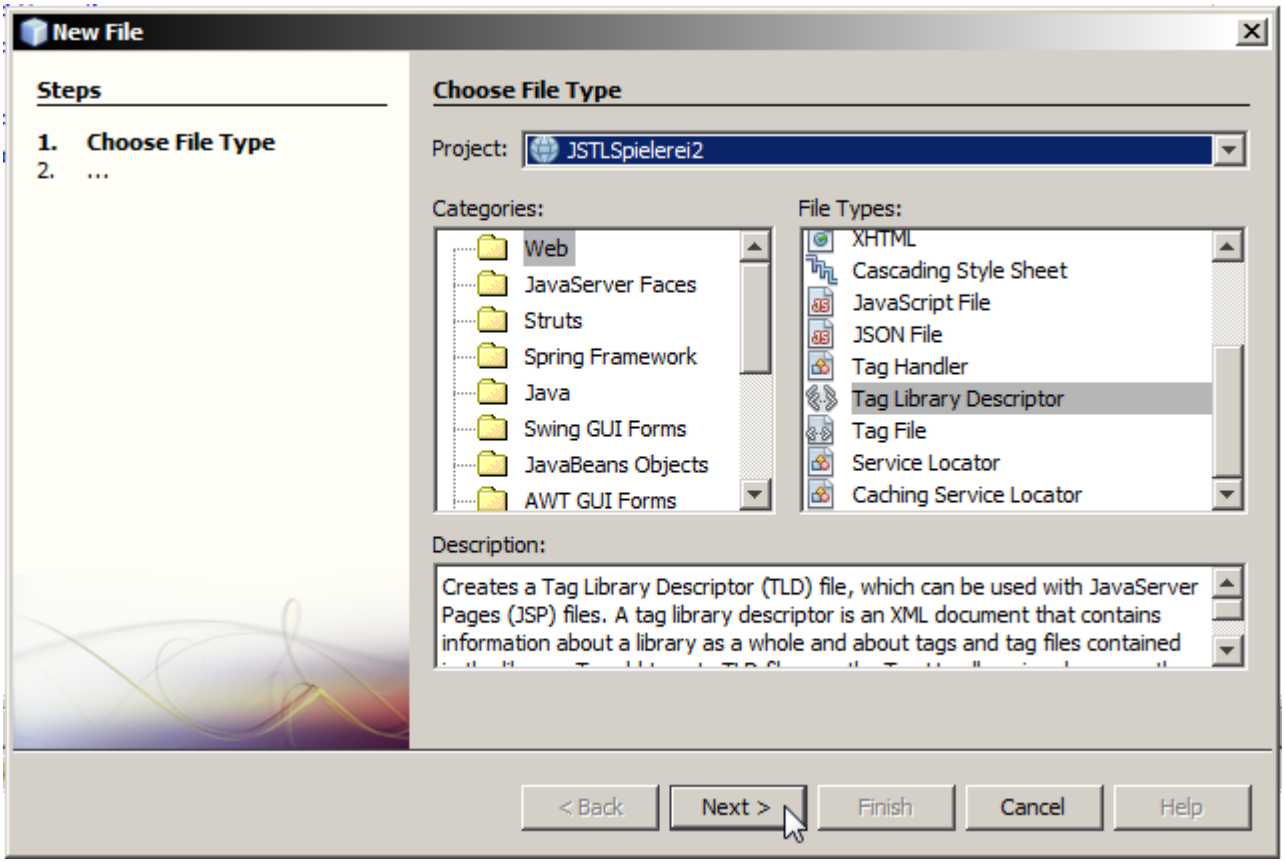
8.9 Erstellung einer eigenen Tag Library

Durch die JSTL kann der Inhalt von JSPs im Wesentlichen frei von Java-Code (Scriptlets) gehalten werden. Trotzdem können eigene Tags sinnvoll sein. Im Folgenden wird ein einfacher Tag erstellt und dessen Nutzung gezeigt. Wieder gilt, dass die Erstellung der benötigten Dateien von Hand nicht sehr aufwändig ist, allerdings die Nutzung von Wizzards hilft Fehler zu vermeiden. Für Anfänger ist es trotzdem wichtig sich selbst zu erklären, welche Datei wozu entsteht.
Der entstehende Tag soll eine ganze Zahl und eine Stellenanzahl übergeben bekommen, und dafür sorgen, dass die Zahl mit genau der angegebenen Stellenanzahl ausgegeben wird. Die Ausgabe wird

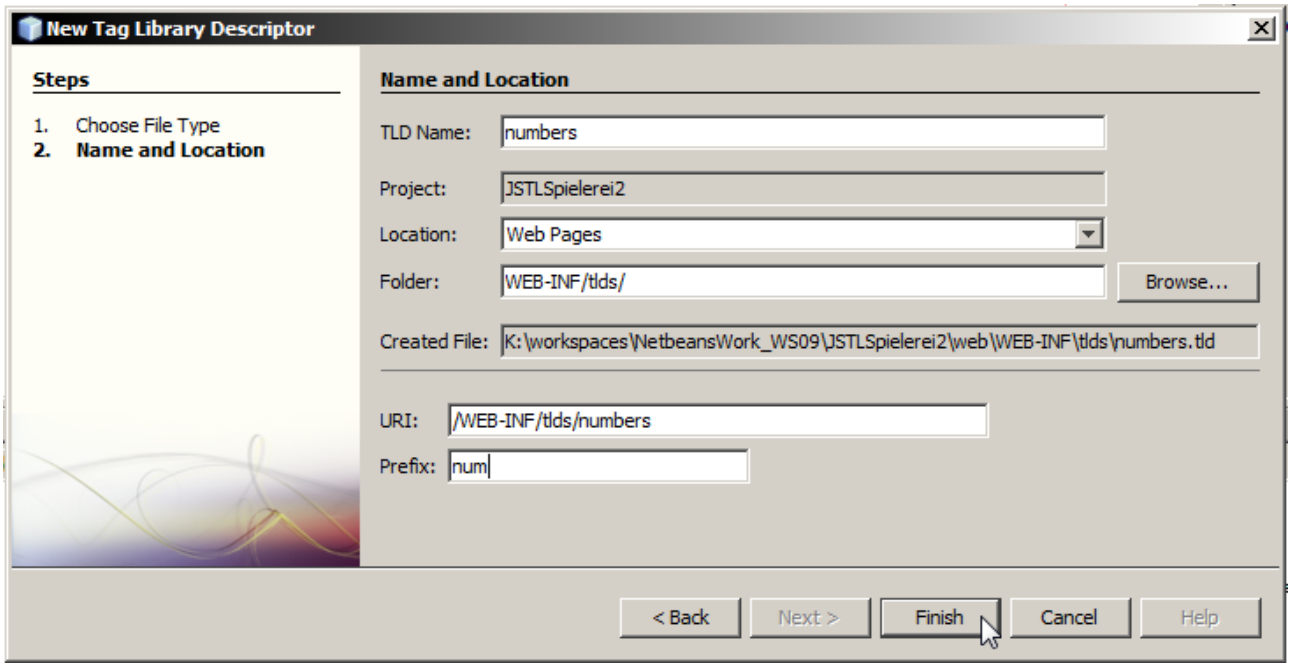
gegebenenfalls mit führenden Nullen aufgefüllt.
Zunächst wird im Projekt ein Rechtsklick z. B. auf dem Ordner „Source Packages“ gemacht und dort „New“ und „Other...“ ausgewählt.



Hier wird unter „Categories“ dann „Web“ und auf der rechten Seite „Tag Library Descriptor“ ausgewählt. (Man beachte, dass in NetBeans erst der Deskriptor und dann der Handler angelegt werden muss.



Nun wird ein Name für den TLD (Tag Library Descriptor) eingegeben. Der vorgeschlagene Speicherort kann übernommen werden, wobei die Platzierungsregeln für den TLD, grob irgendwo im WEB-INF relativ frei sind. Der vorgeschlagene Ansatz ist es, alle TLDs an zentraler Stelle zu sammeln. Dies erleichtert den Überblick über das Projekt wesentlich, macht aber Änderungen im Handler, die in der TLD-Datei nachgezogen werden müssen, etwas fehleranfälliger.
Die „URI“ und der „Prefix“ können übernommen werden, müssen es aber nicht. Im folgenden Beispiel wurde der Präfix etwas gekürzt.



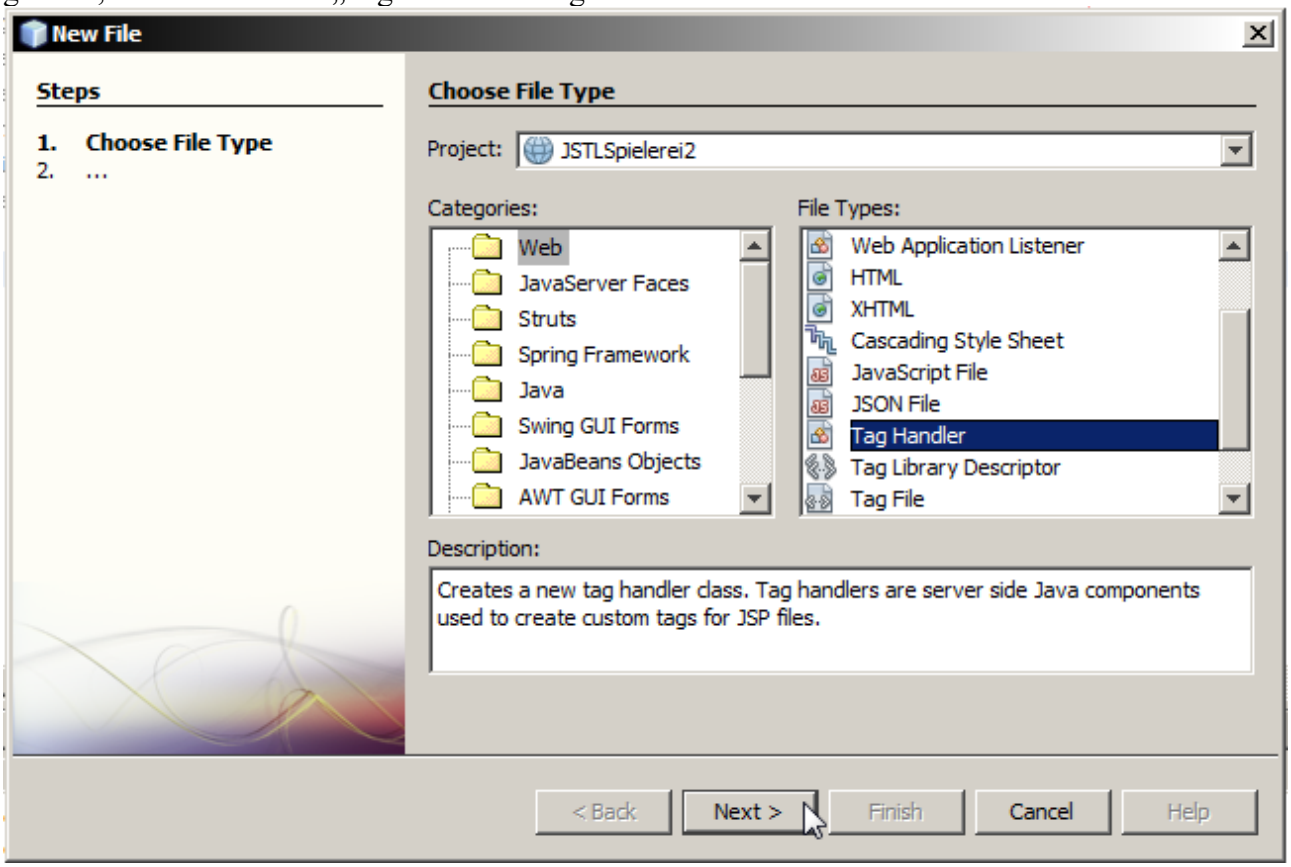
Die entstandene TLD-Datei sieht wie folgt aus und enthält für Anfänger recht interessante Hinweise.

index.jsp × numbers.tld ×

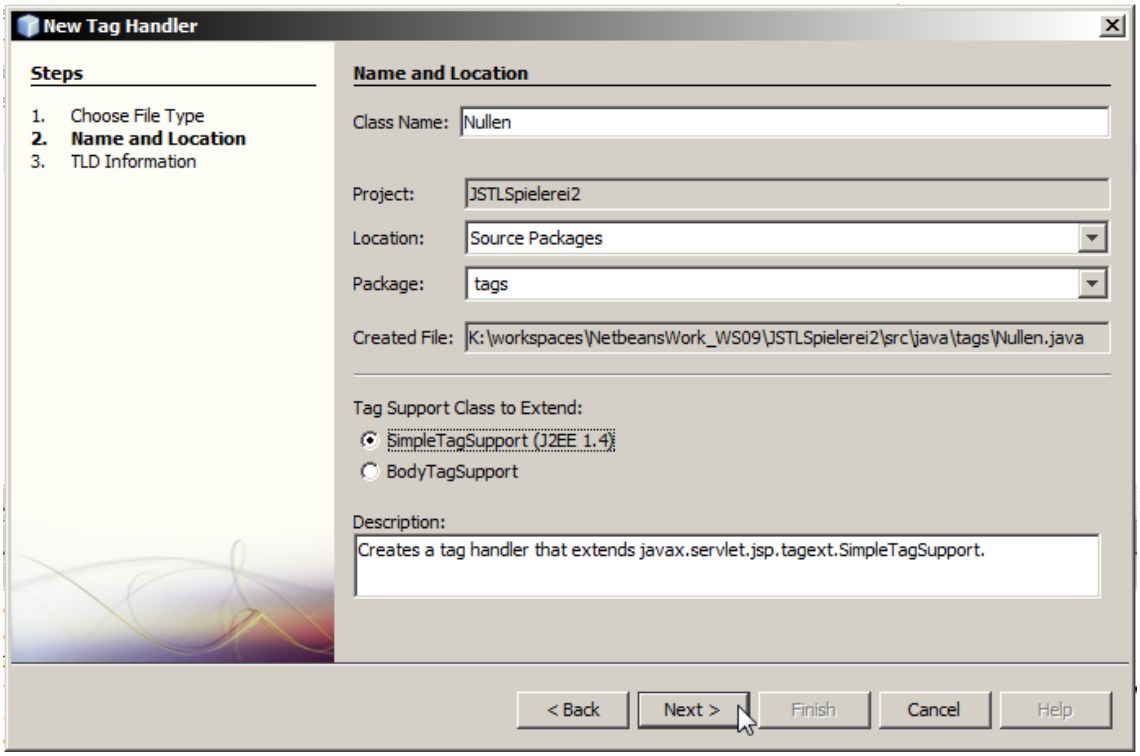
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://
3    <tlib-version>1.0</tlib-version>
4    <short-name>num</short-name>
5    <uri>/WEB-INF/tlds/numbers</uri>
6    <!-- A validator verifies that the tags are used correctly at JSP
7           translation time. Validator entries look like this:
8        <validator>
9            <validator-class>com.mycompany.TagLibValidator</validator-class>
10           <init-param>
11              <param-name>parameter</param-name>
12              <param-value>value</param-value>
13           </init-param>
14       </validator>
15    -->
16   <!-- A tag library can register Servlet Context event listeners in
17         case it needs to react to such events. Listener entries look
18         like this:
19       <listener>
20           <listener-class>com.mycompany.TagLibListener</listener-class>
21       </listener>
22    -->
23 </taglib>
```

Nun soll der Tag-Handler geschrieben werden. Dazu wird der gleiche New-Dialog wie vorher

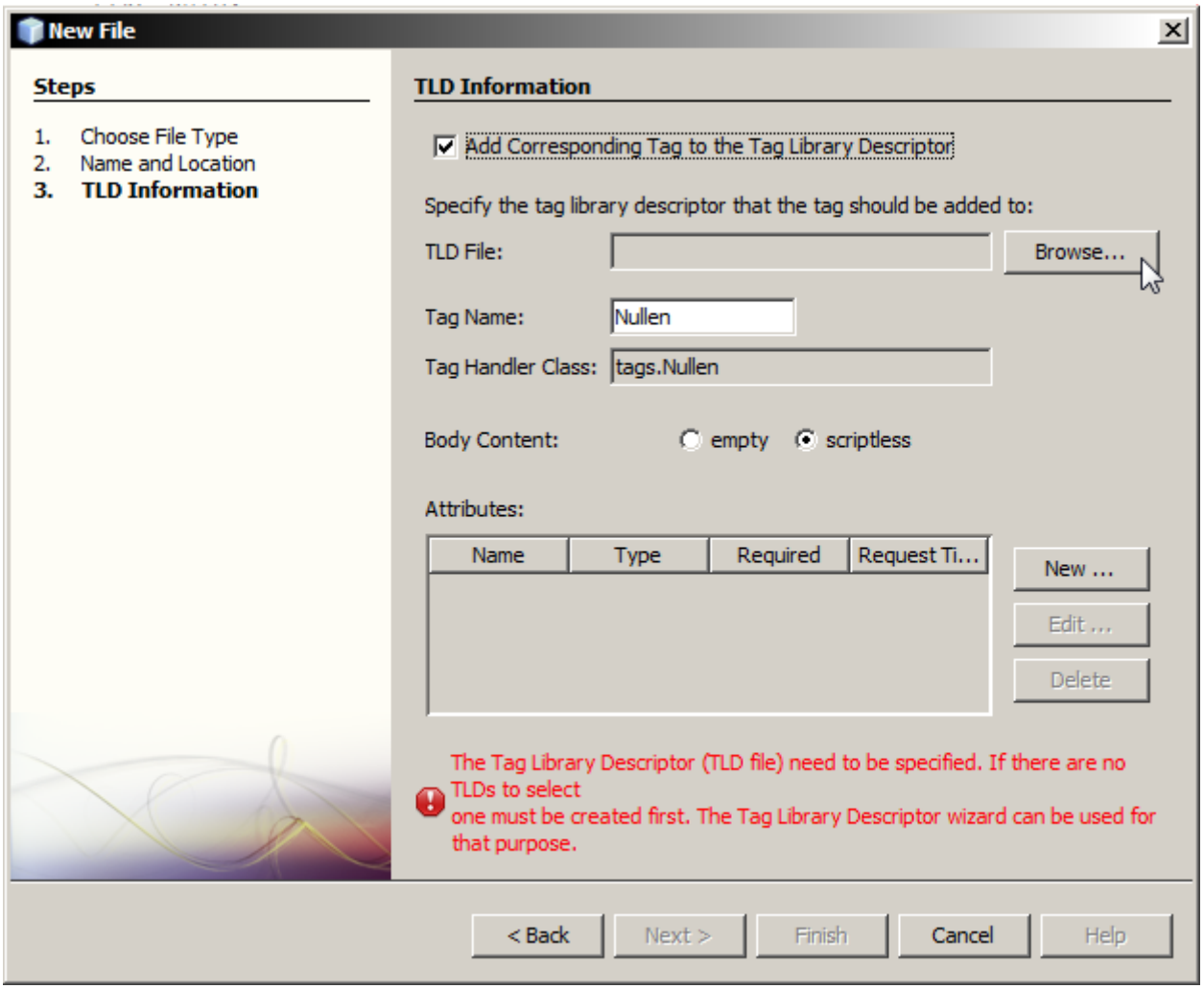
genutzt, dann aber rechts „Tag Handler“ ausgewählt.



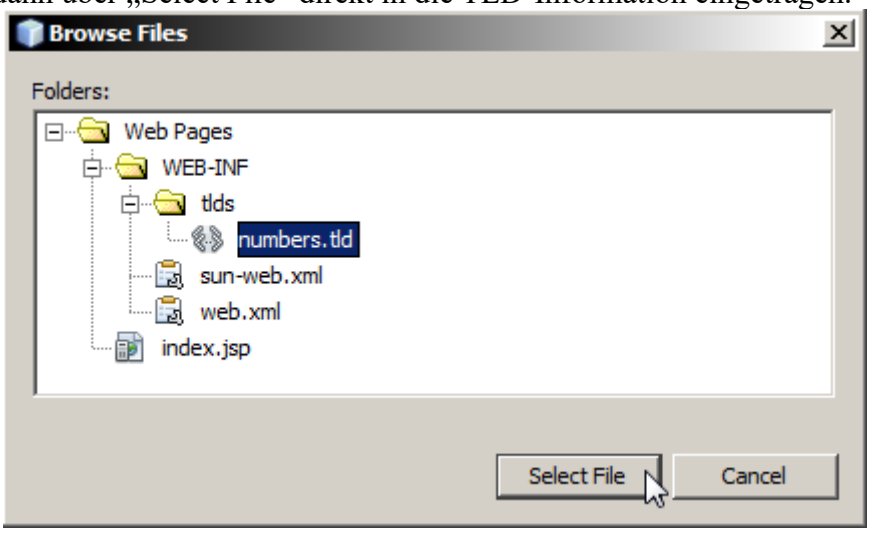
Es muss ein Klassenname „Class name“ vergeben werden, weiterhin ist es sehr sinnvoll, die Implementierung in einem Paket („Package“) abzulegen. Der Rest der Einstellungen ist so zu übernehmen.



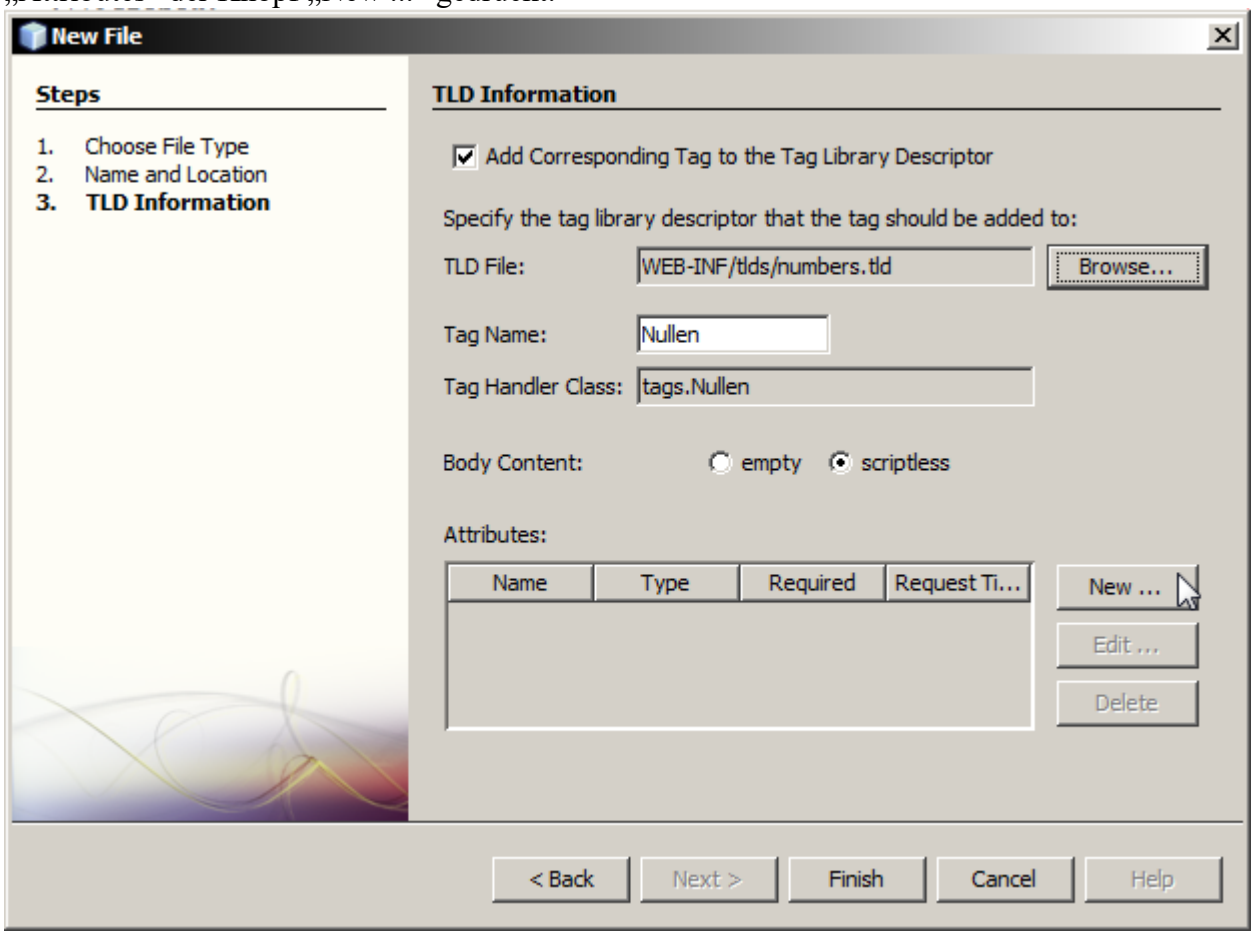
Nun wird zunächst rechts-oben unter „Browse...“ der gerade erstellte TLD gesucht.



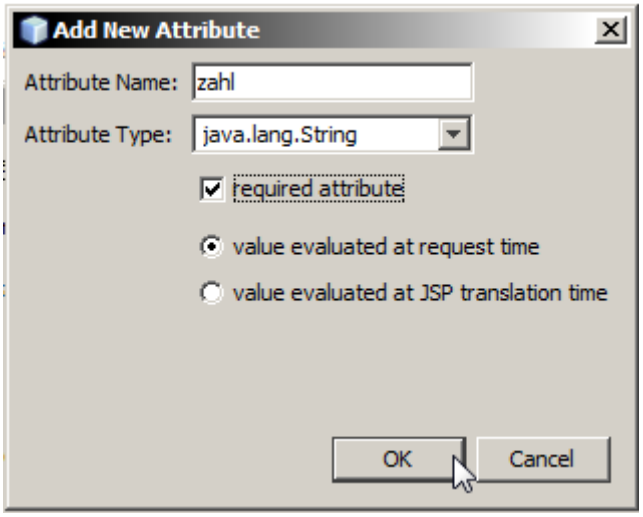
Die Datei wird dann über „Select File“ direkt in die TLD-Information eingetragen.



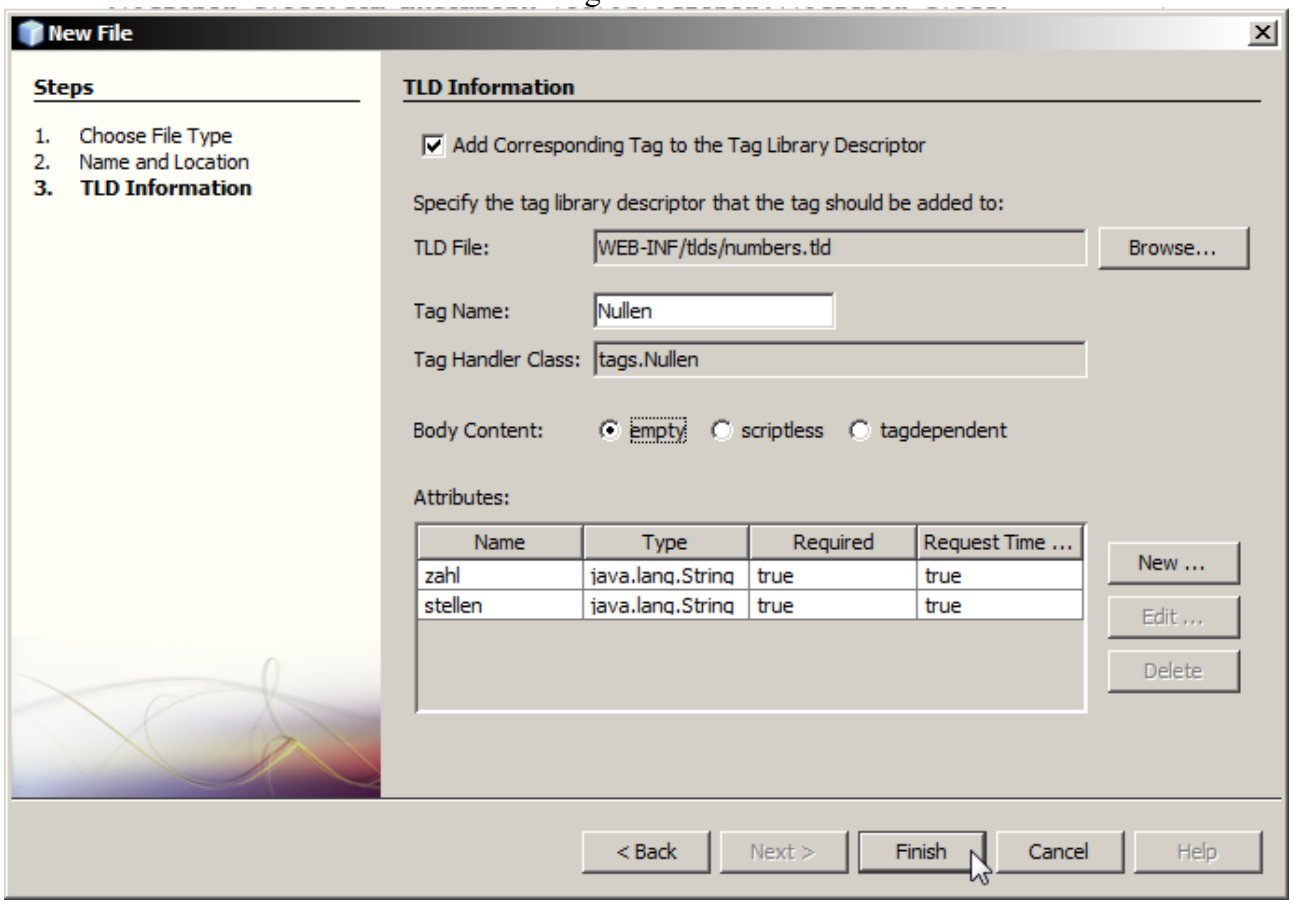
Danach können die Parameter, genauer die Attribute des Tags festgelegt werden. Hierzu wird bei „Attributes“ der Knopf „New ...“ gedrückt.



Da hier einedetaillierte Behandlung von Fehleingaben nicht betrachtet werden soll, werden beide Parameter als Strings übergeben, was bei tiefergehender Betrachtung nicht notwendig ist. Man kann weiterhin angeben, ob das Attribut verpflichtend („required“) ist.



Insgesamt sieht die vollständige TLD-Information für das Beispiel wie folgt aus, dabei wurde der „Body Content“ noch auf „empty“ gesetzt, da die bearbeiteten Werte ausschließlich über die Attribute und keine Inhalte des XML-Elements übertragen werden sollen.



Die resultierende Datei enthält schon alle Exemplarvariablen und Methoden zur Nutzung der Attribute. Weiterhin wird eine Struktur für die zentrale doTag()-Methode vorgeschlagen, die man

ergänzen oder auch ignorieren kann. Da es im Beispiel keinen „Body Content“ geben kann, kann auf diesen Teil hier verzichtet werden. Bevor die Datei gezeigt wird, ist hier noch die Ergänzung in der TLD-Datei numbers.tld angegeben.

```
  <tag>
    <name>Nullen</name>
    <tag-class>tags.Nullen</tag-class>
    <body-content>empty</body-content>
    <attribute>
      <name>zahl</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
    <attribute>
      <name>stellen</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
  </tag>
```

index.jsp × numbers.tld × Nullen.java ×

```
4   */
5
6   package tags;
7
8   import javax.servlet.jsp.JspWriter;
9   import javax.servlet.jsp.JspException;
10  import javax.servlet.jsp.tagext.JspFragment;
11  import javax.servlet.jsp.tagext.SimpleTagSupport;
12
13  /**
14   *
15   * @author x
16   */
17  public class Nullen extends SimpleTagSupport {
18    private String zahl;
19    private String stellen;
20
21      /**
22       * Called by the container to invoke this tag.
23       * The implementation of this method is provided by the tag library developer,
24       * and handles all tag processing, body iteration, etc.
25       */
          @Override
27        public void doTag() throws JspException {
28            JspWriter out = getJspContext().getOut();
29
30            try {
31                // TODO: insert code to write html before writing the body content.
32                // e.g.:
33                //
34                // out.println("<strong>" + attribute_1 + "</strong>");
35                // out.println("    <blockquote>");
36
37                JspFragment f = getJspBody();
38                if (f != null) f.invoke(out);
39
40                // TODO: insert code to write html after writing the body content.
41                // e.g.:
42                //
43                // out.println("    </blockquote>");
44
45            } catch (java.io.IOException ex) {
46                throw new JspException("Error in Nullen tag", ex);
47            }
48      }
49
50    public void setZahl(String zahl) {
51      this.zahl = zahl;
52    }
53
54    public void setStellen(String stellen) {
55      this.stellen = stellen;
56    }
57
58  }
```

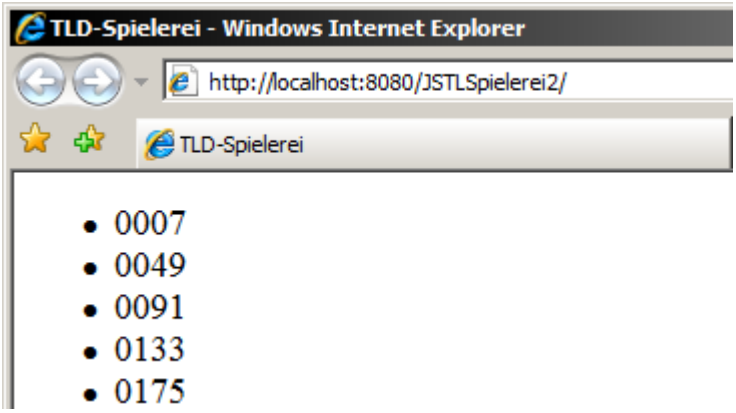
Für das Beispiel wird doTag() wie folgt ausprogrammiert.

```
    @Override
    public void doTag() throws JspException {
        JspWriter out = getJspContext().getOut();
        try {
          int st= Integer.parseInt(stellen);
          int za= Integer.parseInt(zahl);
          StringBuffer ergebnis= new StringBuffer();
          for(int i=0;i<st;i++)
            ergebnis.append("0");
          int position=1;
          while(za>0 && position<=st){
            char c= (char) (48 + (za%10));
            ergebnis.setCharAt(st-position, c);
            position++;
            za=za/10;
          }
          out.println(ergebnis.toString());
          System.out.println(ergebnis);
        } catch (java.io.IOException ex) {
            throw new JspException("Error in Nullen tag", ex);
        }
    }
```

Die Nutzung der neuen Tag-Library erfolgt mit der taglib-Direktive, wie folgendes Beispiel mit zugehöriger Ausgabe zeigt. Hier gibt es keine besondere Unterstützung durch NetBeans.

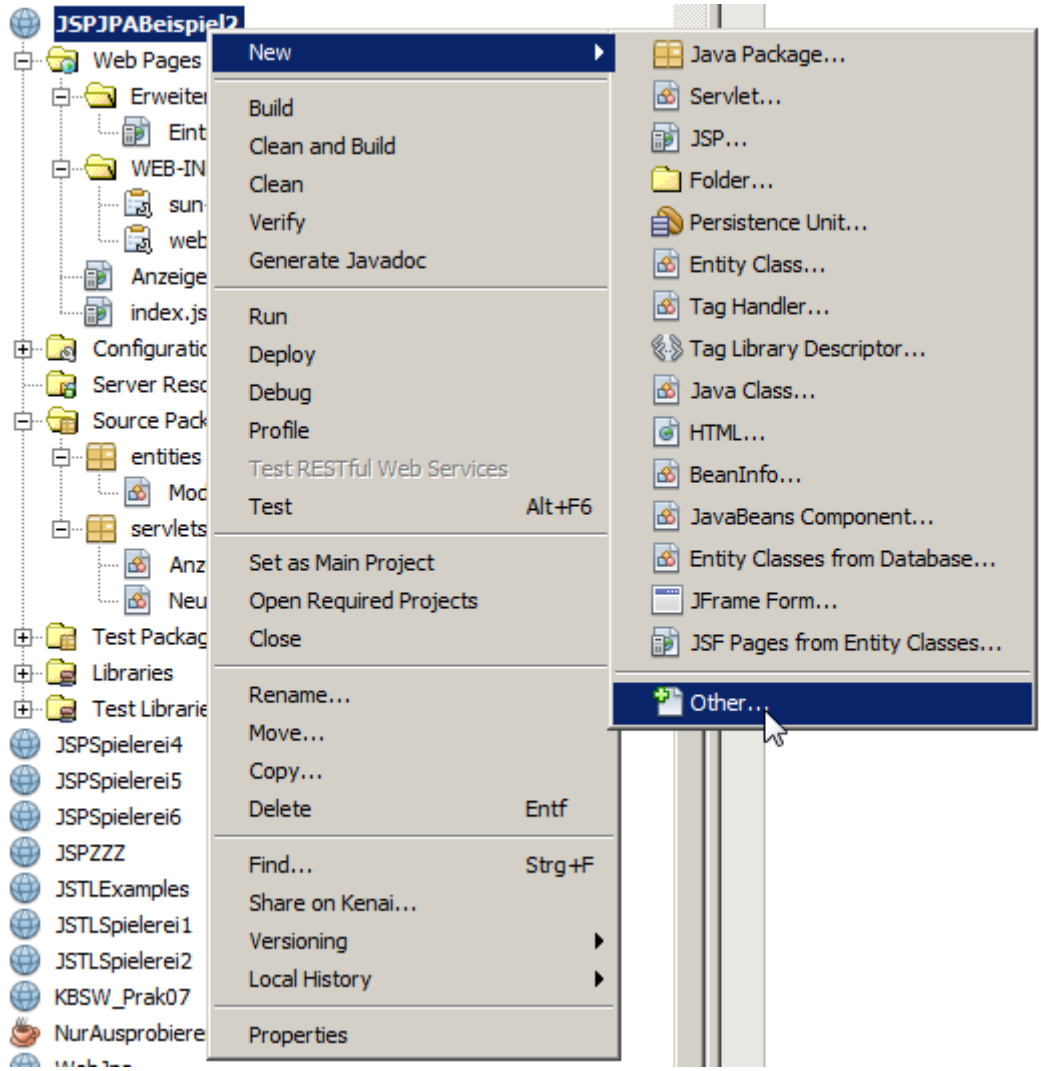
```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="num" uri="/WEB-INF/tlds/numbers" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>TLD Spielerei</title>
  </head>
  <body>
    <ul>
      <c:forEach var="i" begin="7" end="180" step="42">
        <li> <num:Nullen zahl="${i}" stellen="4"/> </li>
      </c:forEach>
    </ul>
  </body>
</html>
```

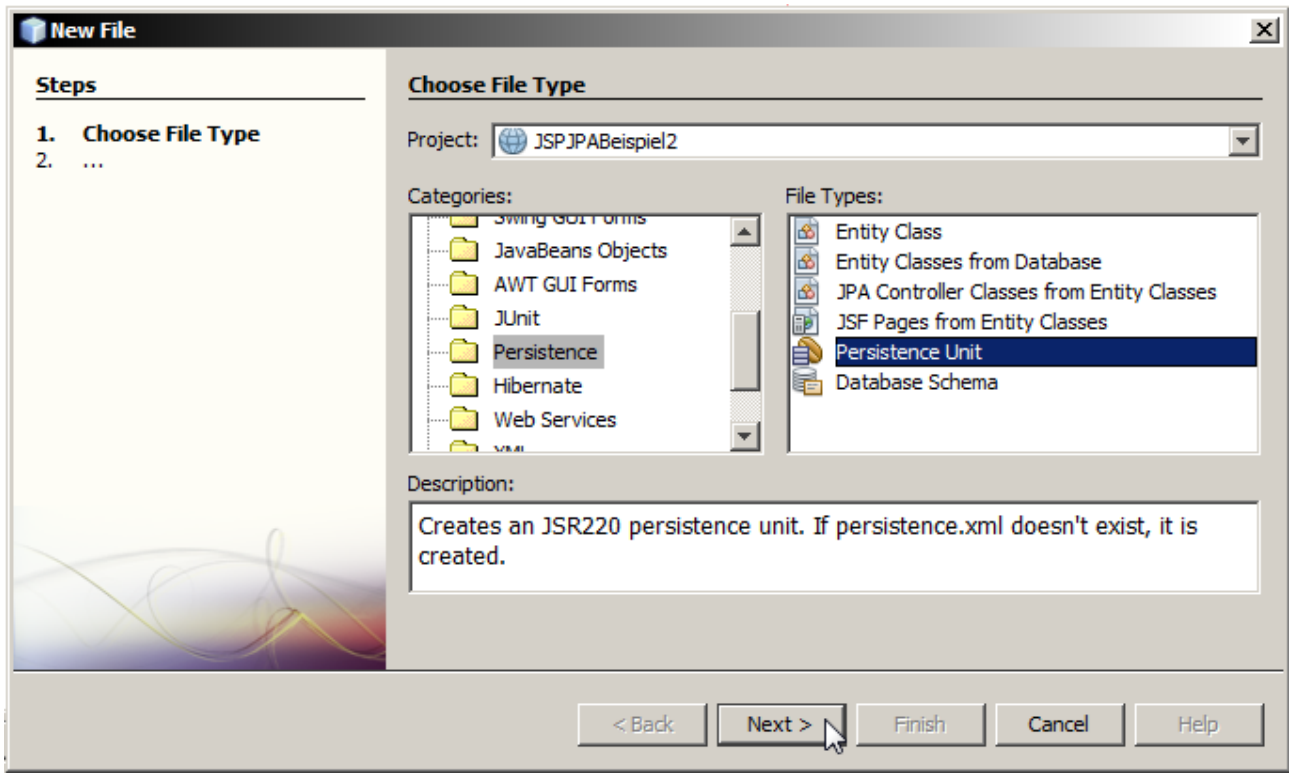


8.10 JSP mit Datenbanken

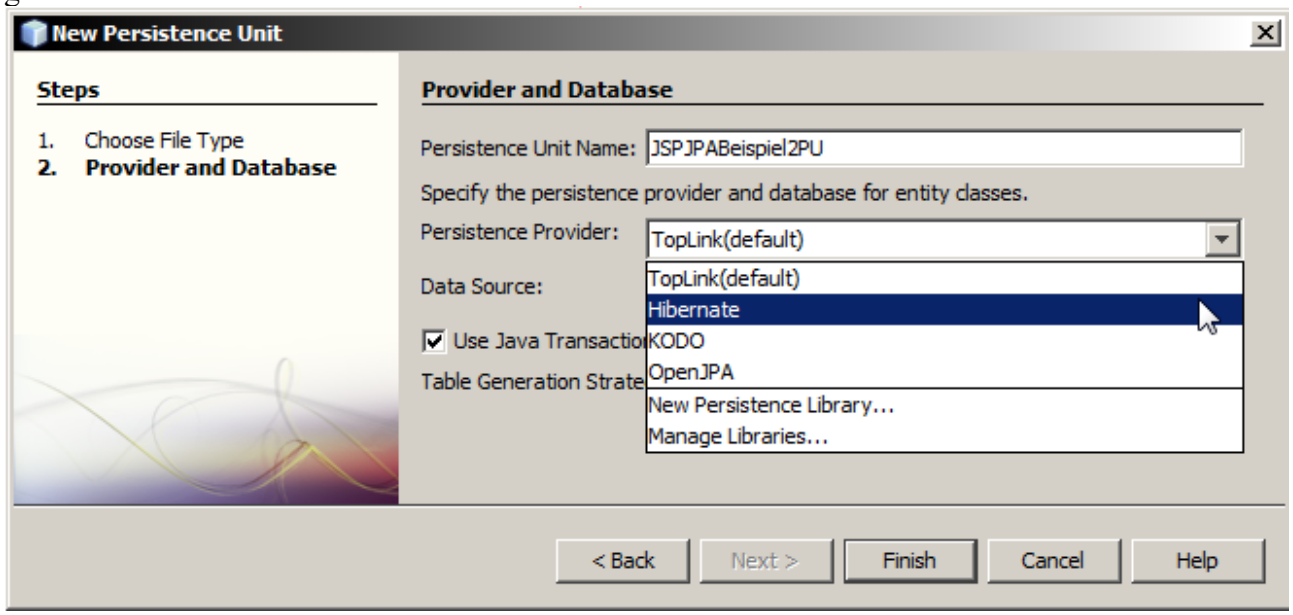
In diesem Abschnitt wird gezeigt, wie man eine Datenbank mit Servlets und JSP nutzen kann, die Kenntnisse des Kapitel „6 NetBeans und Datenbanken“ werden vorausgesetzt.
Gegeben sei eine angelegte Datenbank jdbc:derby://localhost:1527/JSPJPASpielerei2.
Weiterhin gibt es ein Web-Projekt JSPJPASpielerei2, zu dem nachtäglich jetzt eine DB-Unterstützung eingebaut werden soll. Die DB-Nutzung kann auch bereits bei der Projekterstellung angegeben werden, wobei hier teilweise für den OR-Mapper individuelle Dateien angelegt werden. Der wichtige Schritt mit der „Data Source“ ist aber auch hier zu beachten.
Um die DB-Nutzung zu ergänzen, wird zunächst ein Rechtsklick auf dem Projekt gemacht und „New“ sowie „Other...“ gewählt.



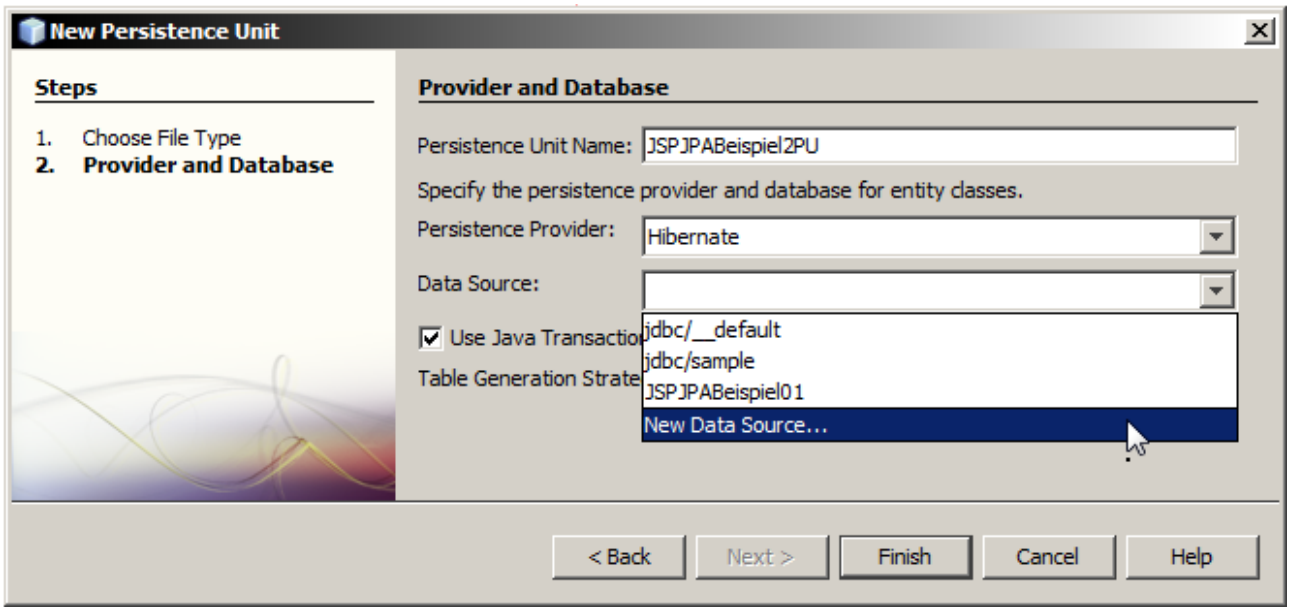
Danach wird links „Persistence“ und rechts „Persistence Unit“ ausgewählt.



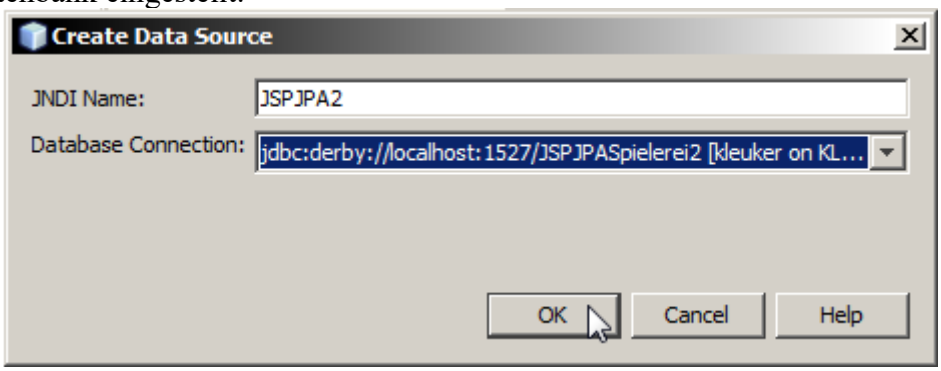
Der Name der Persistence Unit kann beibehalten werden, der „Provider“ wird auf „Hibernate“ geändert.



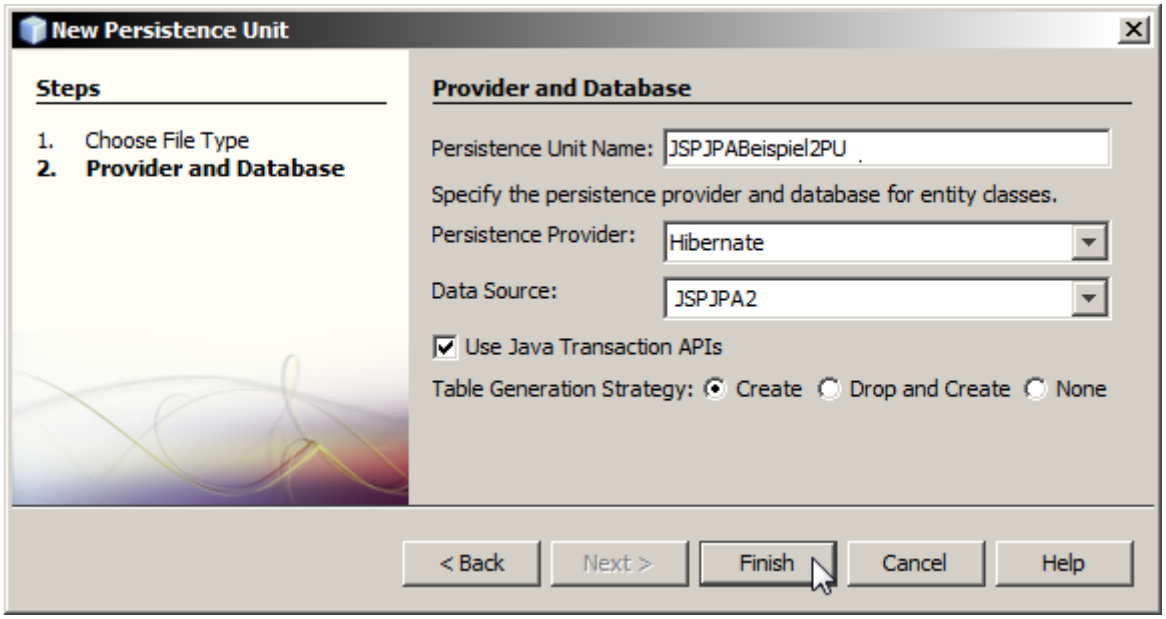
Jetzt wird der Pfeil rechts von „Data Source“ angeklickt und „Create New Datasource“ ausgewählt. Mit dem folgenden Schritt können wir unsere vorher angelegte Datenbank als Data Source des Servers registrieren, so dass im Server ein direkter Zugriff über JNDI-Nutzung möglich wird. Diese Schritte werden später intern gemacht.



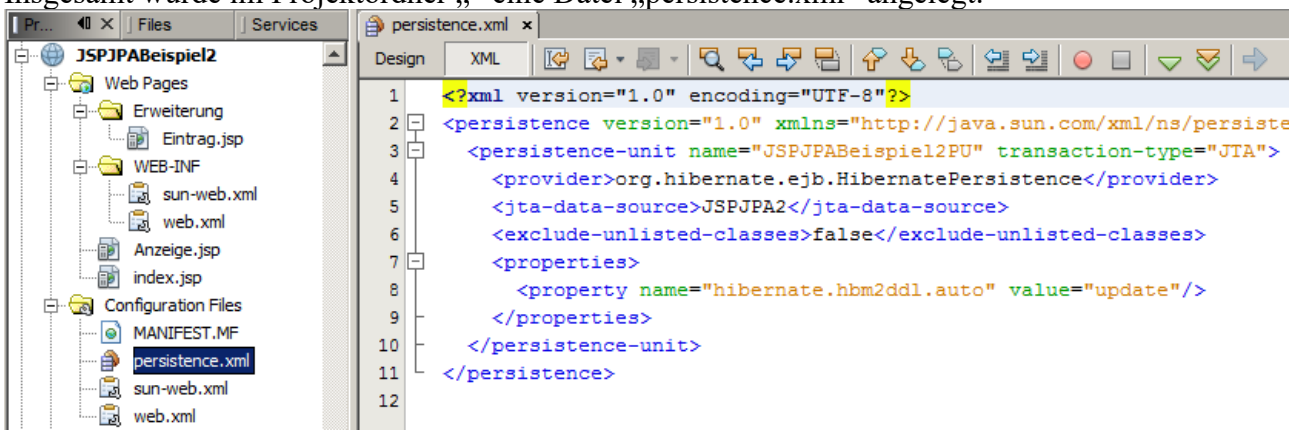
Der JNDI-Name kann relativ frei gewählt werden. Bei der „Database Connection“ wird die vorher angelegte Datenbank eingestellt.



Danach wird das Anlegen der Persistence Unit mit Finish abgeschlossen.



Insgesamt wurde im Projektordner „“ eine Datei „persistence.xml“ angelegt.



Danach können Entitäten, wie bei der JPA-Nutzung beschrieben, angelegt werden. Die Besonderheit ist, dass der Container die gesamte Transaktionssteuerung, inklusive benötigter Rollbacks, übernimmt. Im Programm wird dazu ein spezielles UserTransaction–Objekt genutzt, wie der folgende Beispiel-Code zeigt.

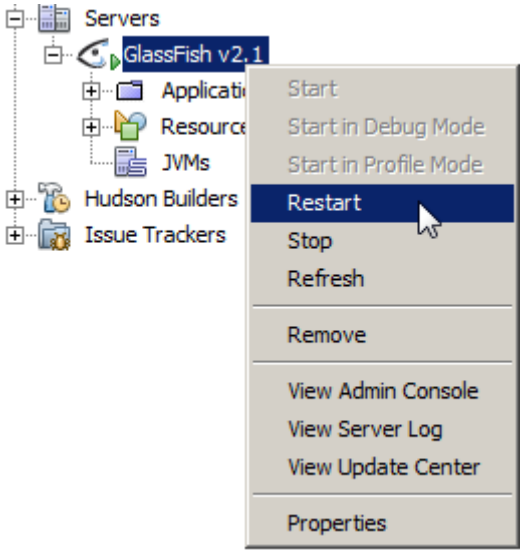
```
public class NeuesModul extends HttpServlet {

  @PersistenceUnit
  private EntityManagerFactory emf;
  @Resource
  private UserTransaction utx;

  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try {
      utx.begin();
      EntityManager em = emf.createEntityManager();
```

```
            Modul m = new Modul(Integer.parseInt(request.getParameter("nr")),
                    request.getParameter("name"));
            em.persist(m);
            utx.commit();
            request.getRequestDispatcher("/index.jsp").forward(request, response);
        } catch (RollbackException ex) {
            Logger.getLogger(NeuesModul.class.getName()).log(Level.SEVERE, null, ex);
        } catch (HeuristicMixedException ex) {
            Logger.getLogger(NeuesModul.class.getName()).log(Level.SEVERE, null, ex);
        } catch (HeuristicRollbackException ex) {
            Logger.getLogger(NeuesModul.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SecurityException ex) {
            Logger.getLogger(NeuesModul.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IllegalStateException ex) {
            Logger.getLogger(NeuesModul.class.getName()).log(Level.SEVERE, null, ex);
        } catch (NotSupportedException ex) {
            Logger.getLogger(NeuesModul.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SystemException ex) {
            Logger.getLogger(NeuesModul.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

Ein zentraler Schritt bei der Evaluation, dass die DB-Nutzung funktioniert, ist der Restart des Servers, nach dem die Daten weiterhin vorhanden sein sollten. Der Restart kann direkt in NetBeans durchgeführt werden.



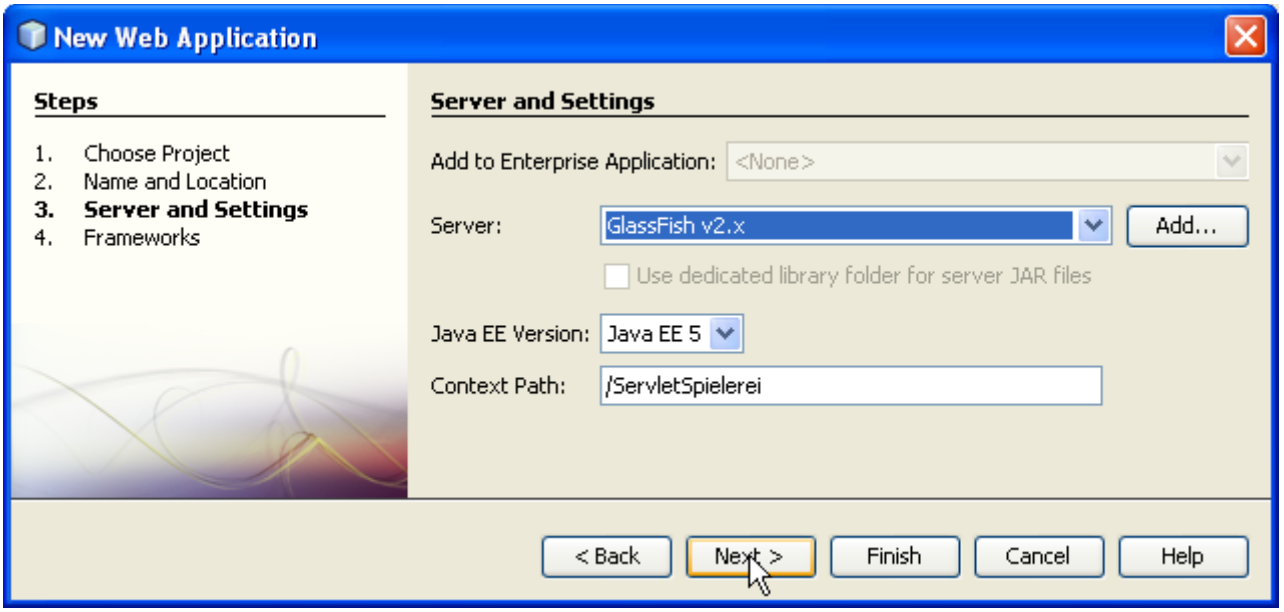
Man beachte, dass durch eine Programmänderung die Applikation wieder neu deployed wird, wodurch alle Daten verloren gehen. In der Realität würde man zunächst z. B. ein Backup der Datenbank durchführen und die Daten nach dem Deployen wieder einspielen.

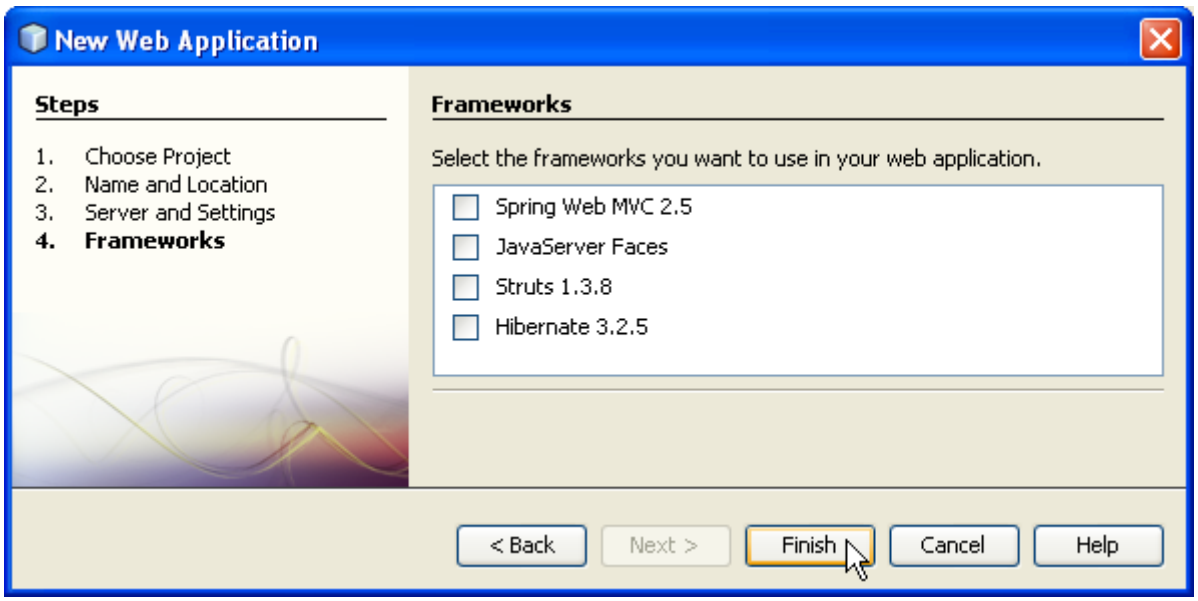
9 Servlets (NetBeans 6.7)

Eine reine Servlet-Entwicklung wird von NetBeans nicht direkt unterstützt, da diese sehr unüblich ist. Üblich ist die Kombination mit JSPs als Frontend und Servlets als Controller. In diesem Kapitel wird die direkte Entwicklung mit Servlets beschrieben, die Kombination mit JSPs kann daraus abgeleitet werden, da diese einfach durch gegenseitige Aufrufe verknüpft werden.

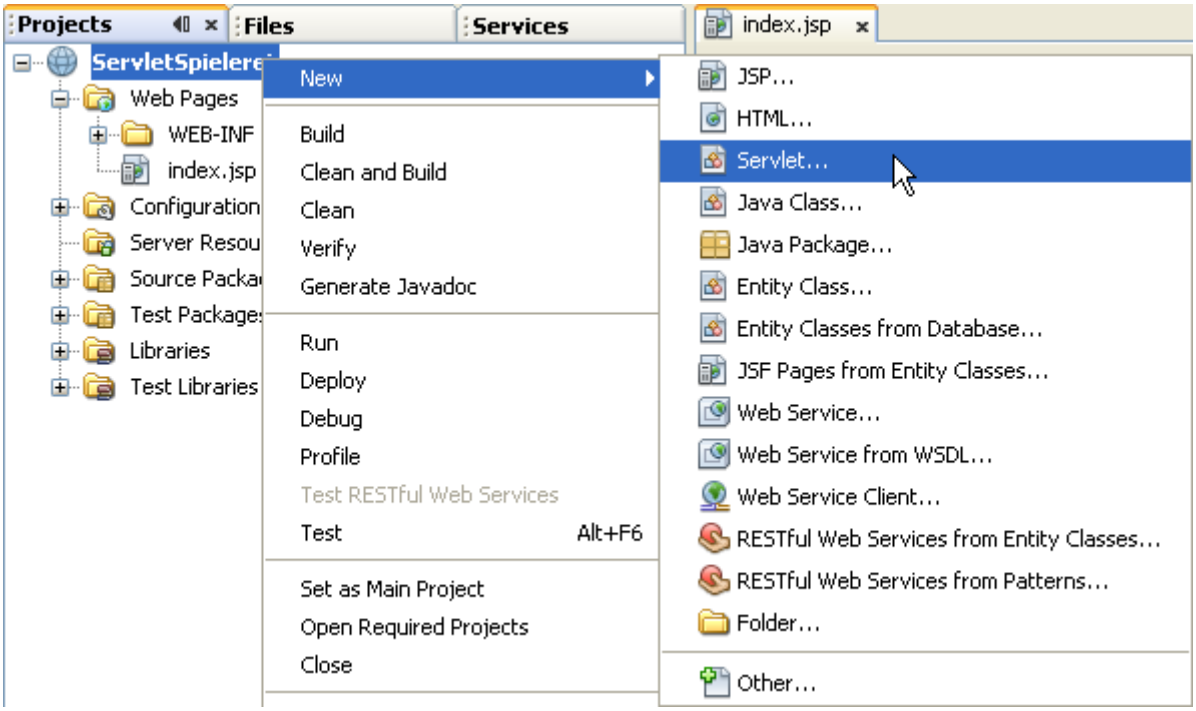
9.1 Anlegen eines Servlets

Zunächst wird ein Web-Projekt ServletSpielerei, wie in Kapitel 8 beschrieben, angelegt, es werden allerdings keine Frameworks ergänzt.

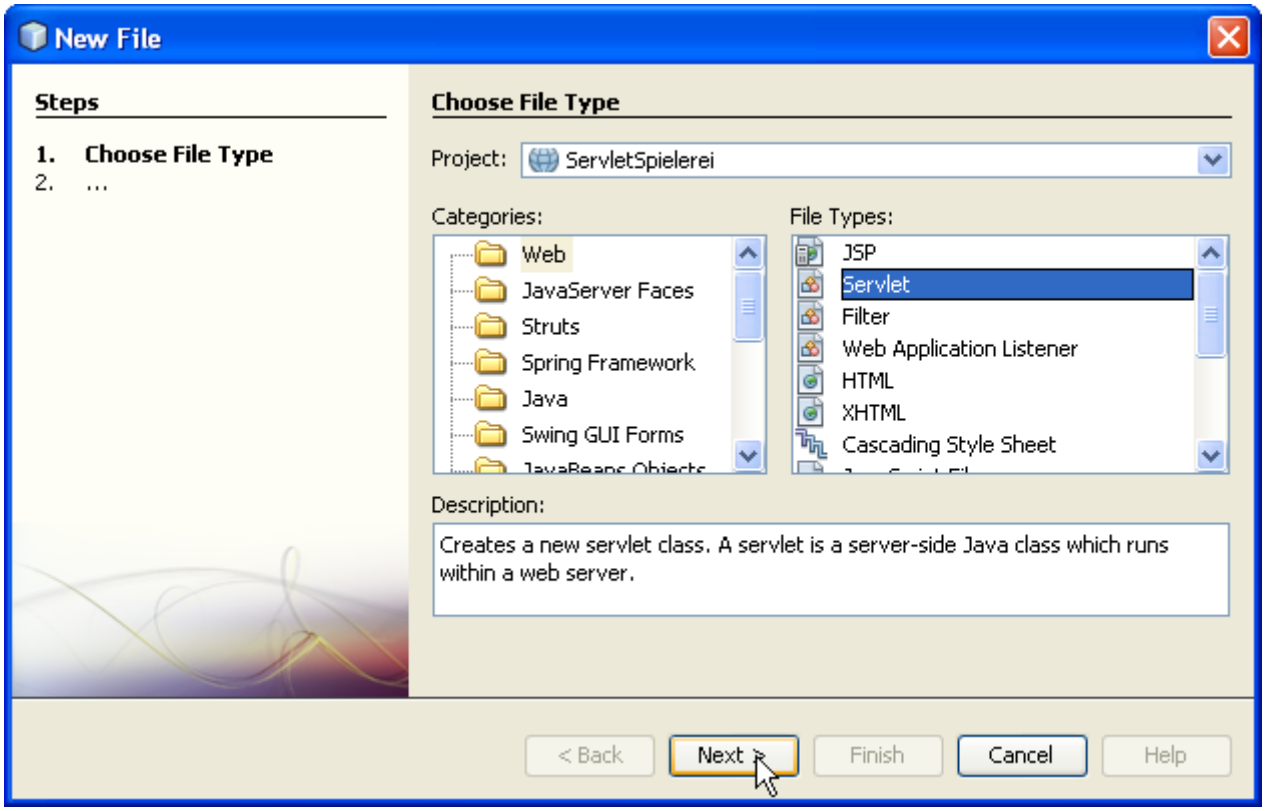




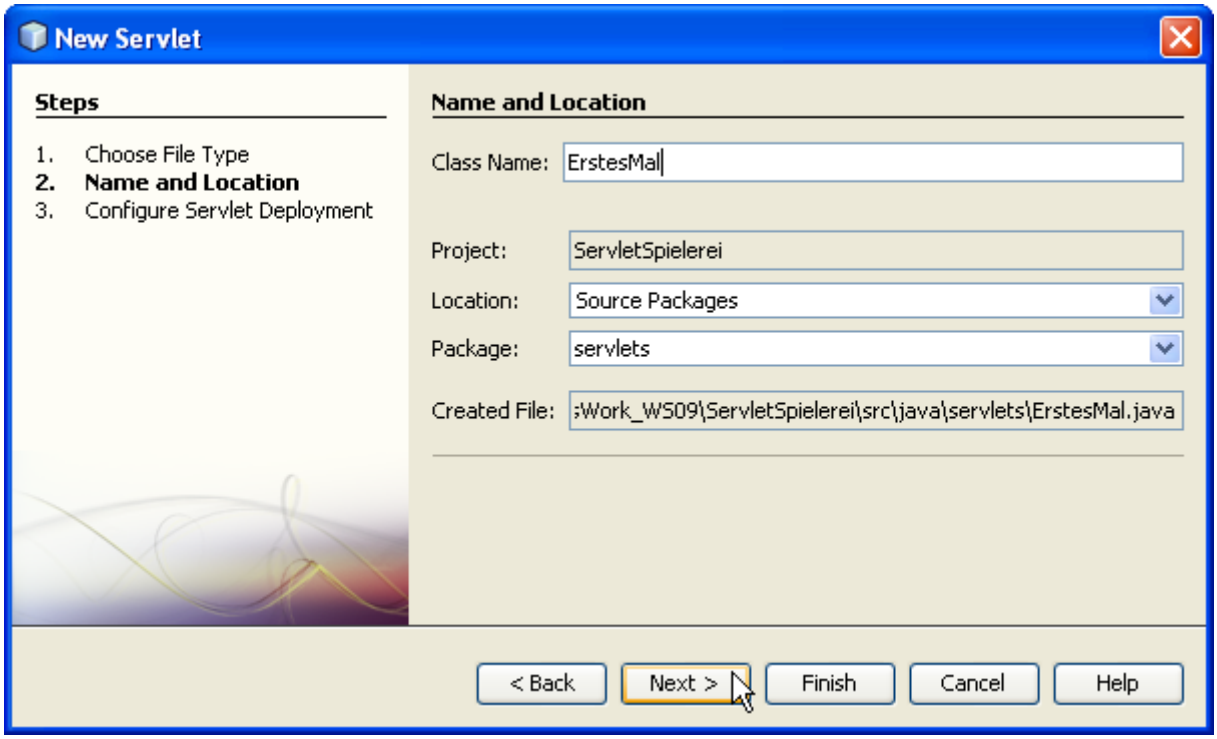
Um ein Servlet anzulegen, wird ein Rechtklick auf dem Projekt gemacht, dann „New“ und „Servlet…“ gewählt.



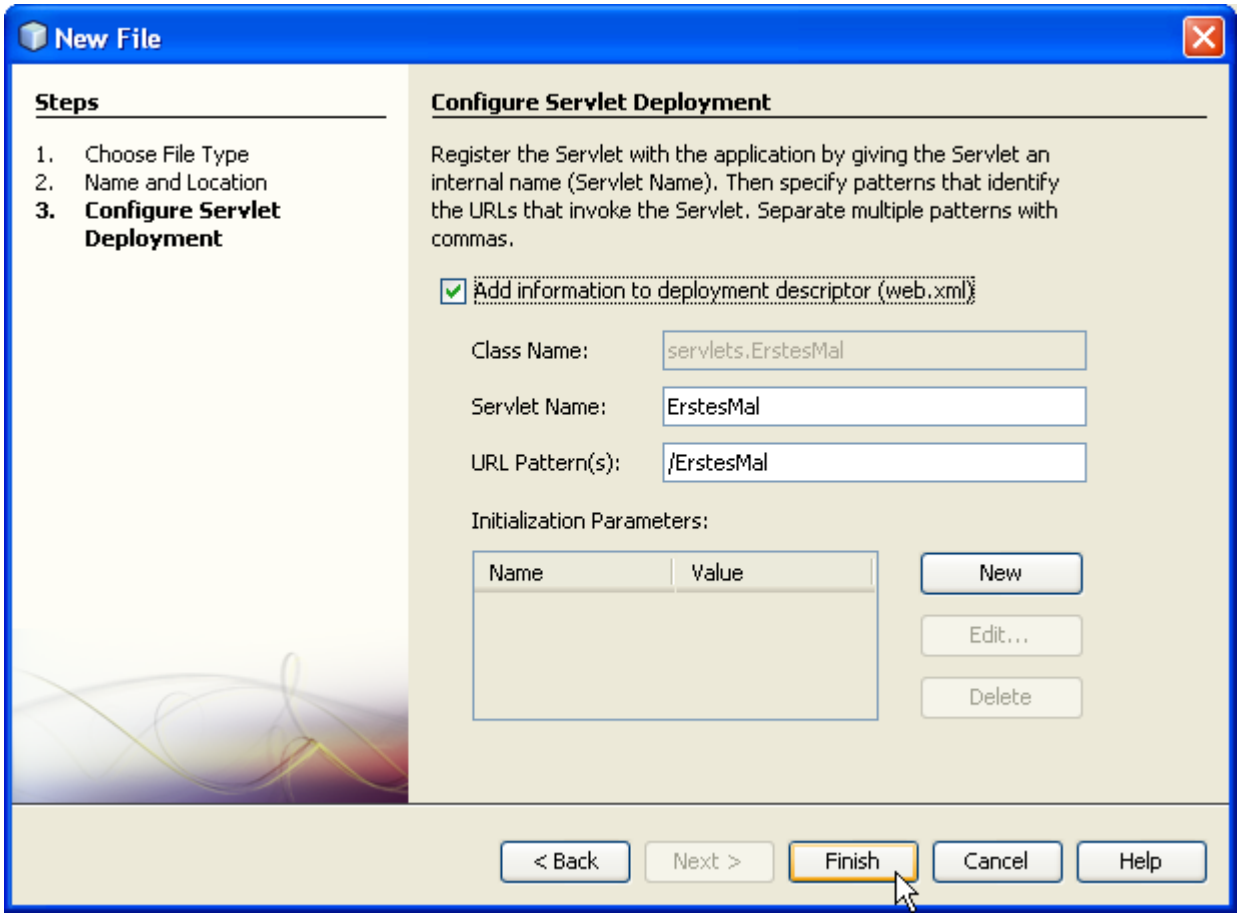
Falls man Servlet nicht direkt finden kann, muss unten „Other…“ gewählt und dann unter „Categories“ als „File Type“ Servlet gewählt werden.



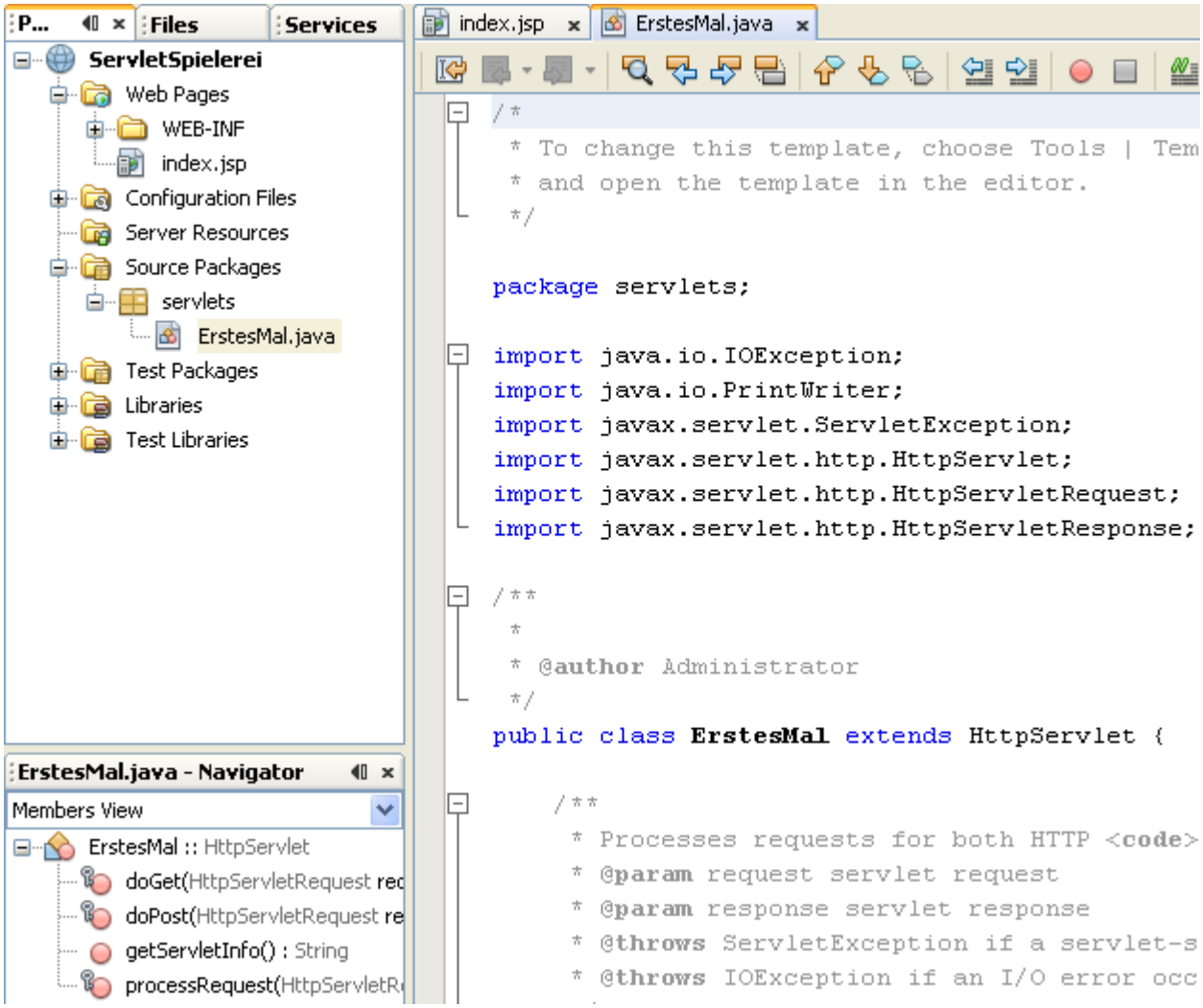
Danach wird ein Klassenname für das neue Servlet eingetragen. Wichtig ist, dass Servlets immer in Paketen, hier „/servlets“, liegen sollten. Man kann danach „Next>“ für weitere Einstellungen wählen.



Die folgenden Einstellungen kann man übernehmen. Man erkennt, dass die Informationen zum Servlet in der zentralen Datei web.xml für Webprojekte abgelegt wird. Grundsätzlich sind der Servlet-Name und das URL-Pattern frei wählbar.



Das Servlet wurde nun mit etwas Code angelegt und kann weiter bearbeitet werden.



9.2 Ausführen eines Servlets

Im folgenden Beispiel wird minimaler Beispielcode ergänzt, genauer die Kommentare um den Beispielcode von NetBeans entfernt. Man beachte, dass in der Methode processRequest die beiden Methoden doGet und doPost zusammengefasst abgearbeitet werden.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet ErstesMal</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet ErstesMal at " + request.getContextPath () + "</h1>");
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}

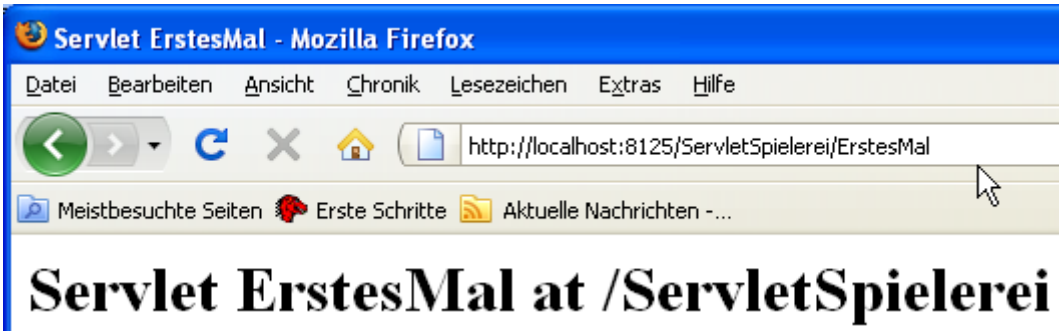
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}
```

Startet man das Projekt jetzt, sieht man von dem Servlet nichts, da durch die Projekteinrichtung eine JSP-Seite für den Start genutzt wird. Bei einer Standardinstallation wird der Port 8080 statt wie 8125 in der folgenden Abbildung genutzt werden.



Statt hier weiter zu konfigurieren, wird einfach manuell im Browser das Servlet aufgerufen.



9.3 Minimales Servlet-Projekt

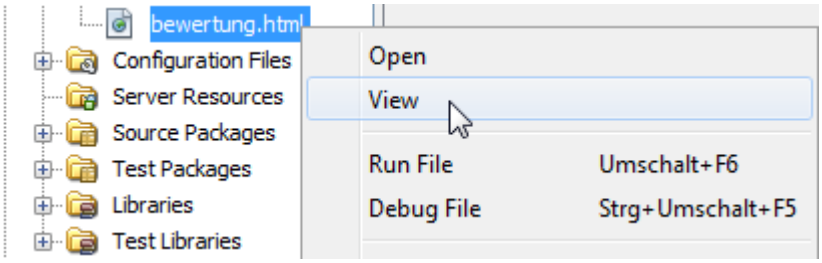
Für ein vollständiges Beispiel wird eine HTML-Seite benötigt, die auf ein Servlet zugreift. Dazu wird folgende Datei bewertung.html angelegt.

web.xml | bewertung.html

```
<!--
To change this template, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Bewertung</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
      <form name="auswertungsform" action="Auswertung" method="POST">
          Verteilte ID eingeben: <input type="text" name="idFeld" value="" size="20" />
          <br>
          <input type="checkbox" name="urteil" value="ausgezeichnet" checked="checked" />
          ausgezeichnet <br>
          <input type="checkbox" name="urteil" value="faszinierend" checked="checked" />
          faszinierend <br>
          <input type="checkbox" name="urteil" value="solala" />
          solala <br>
          <input type="submit" value="Abschicken" name="abschicken" />
      </form>

  </body>
</html>
```

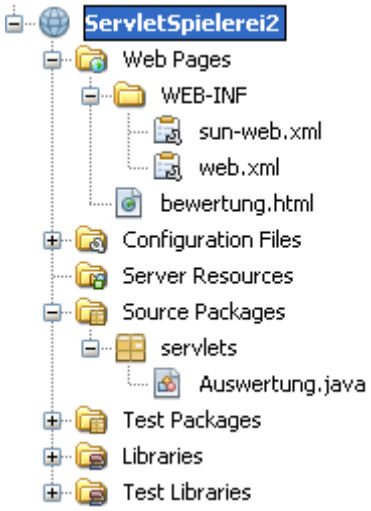
Möchte man eine zwischenzeitlich erstellte HTML-Seite ansehen, so kann man über einen Rechtsklick „View“ auswählen. Die HTML-Datei wird dann im Standard-Browser ausgegeben.



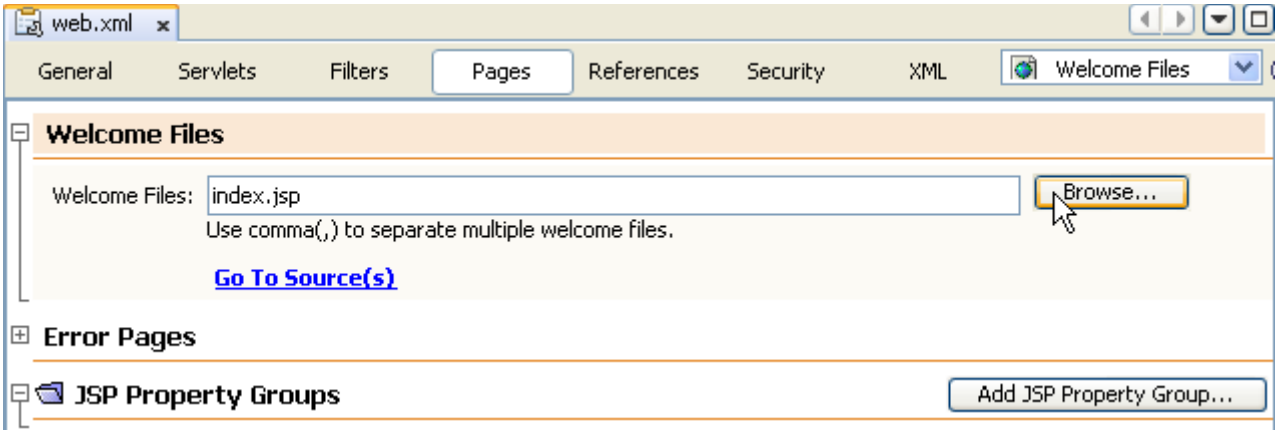
Im nächsten Schritt wird ein Servlet Auswertung.java der folgenden Form angelegt.

```
19  public class Auswertung extends HttpServlet {
20
21      /**
22       * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
23       * @param request servlet request
24       * @param response servlet response
25       * @throws ServletException if a servlet-specific error occurs
26       * @throws IOException if an I/O error occurs
27       */
28      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
29      throws ServletException, IOException {
30          response.setContentType("text/html;charset=UTF-8");
31          PrintWriter out = response.getWriter();
32          String id = request.getParameter("idFeld");
33          String [] werte = request.getParameterValues("urteil");
34          try {
35              out.println("<html> <head>");
36              out.println("<title>Bewertungsergebnisse</title>");
37              out.println("</head>");
38              out.println("<body>");
39              out.println("<h3>Auswertung mit ID: "
40                      + id + "</h3><ul>");
41              for(int i=0;i<werte.length;i++)
42                out.println("<li>"+werte[i]+"</li>");
43              out.println("</ul></body></html>");
44          } finally {
45              out.close();
```

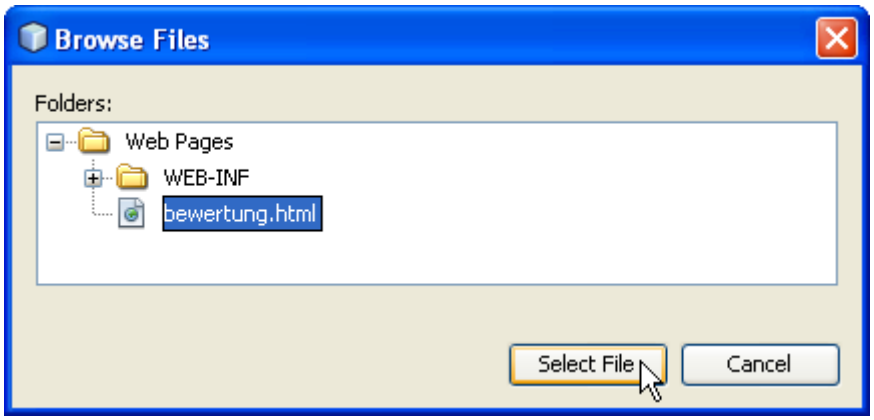
Das zugehörige Projekt hat dabei die folgende Form.



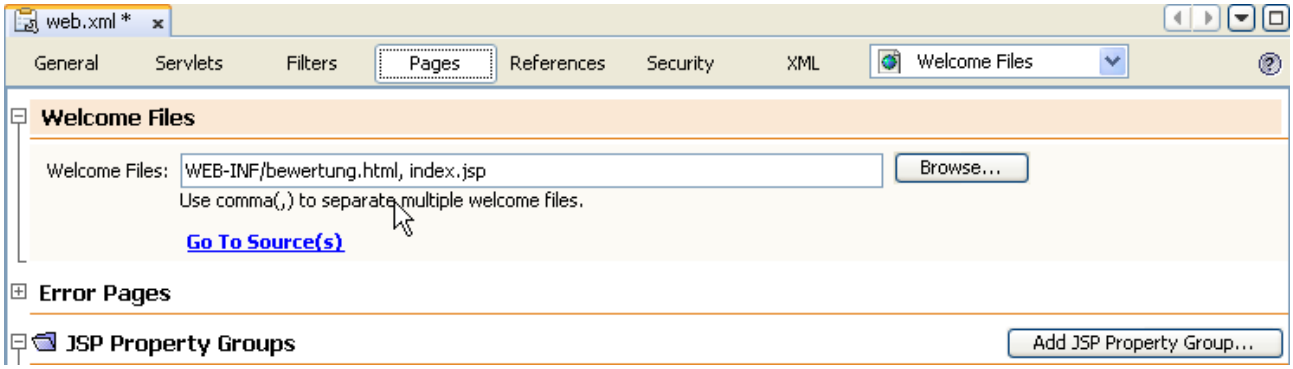
Weiterhin muss die Datei web.xml bearbeitet werden. Unter Pages wird rechts Browse ausgewählt.



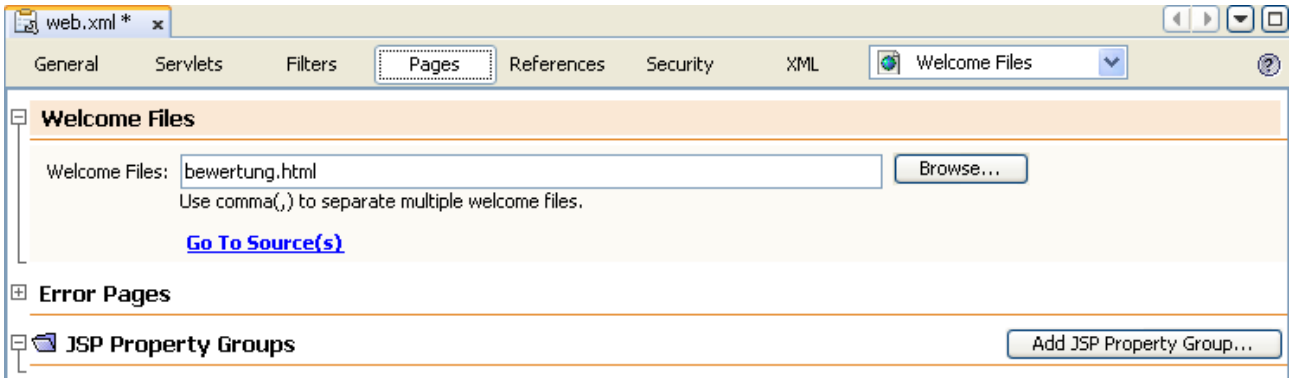
Danach wird zur erstellten HTML-Seite manövriert und diese ausgewählt.



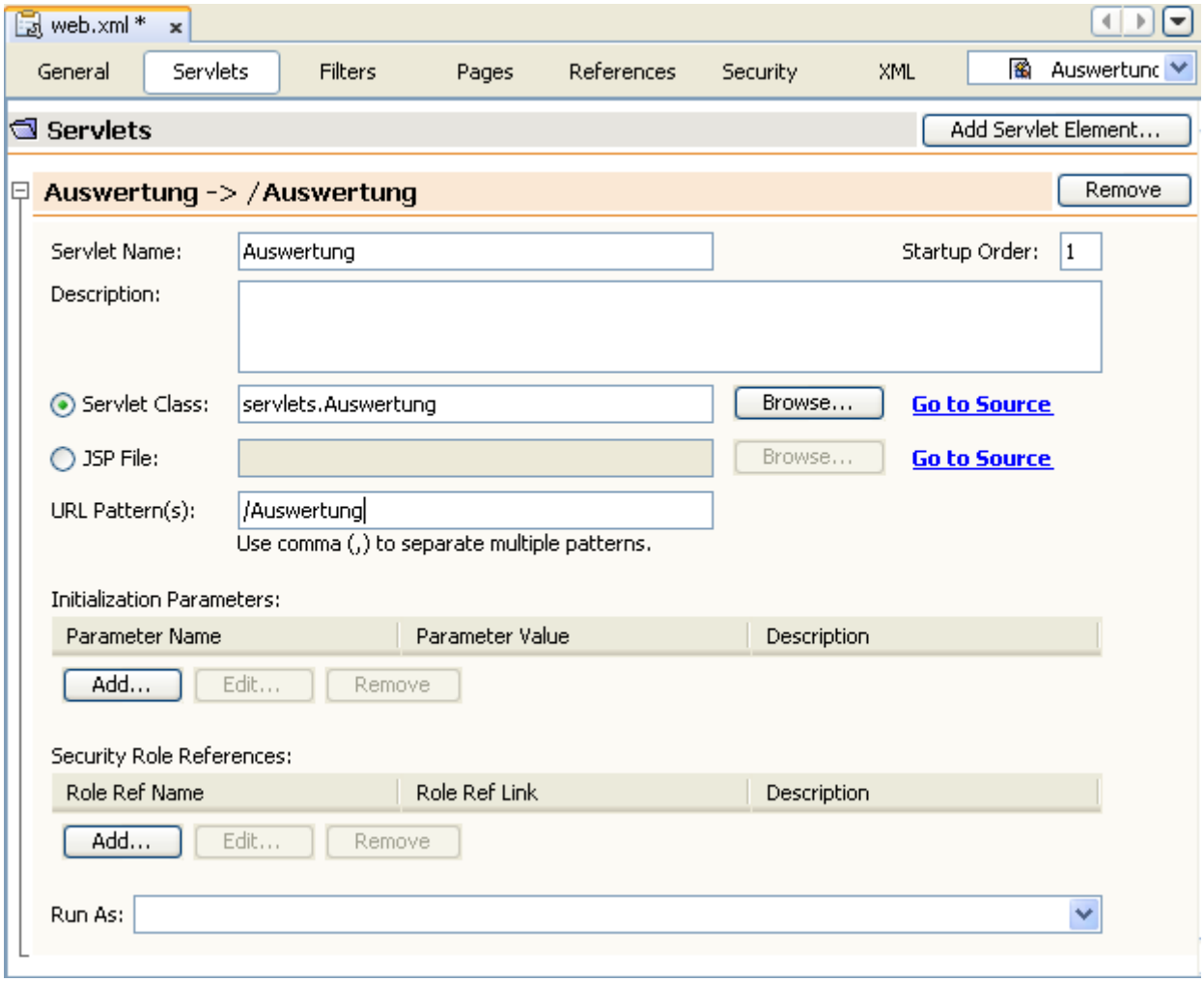
Danach wird der Eintrag zur Startseite so geändert, dass nur bewertung.html zum Start genutzt wird.



Die neue Einstellung sieht wie folgt aus.



Unter Servlets muss das entwickelte Servlet eingestellt sein.



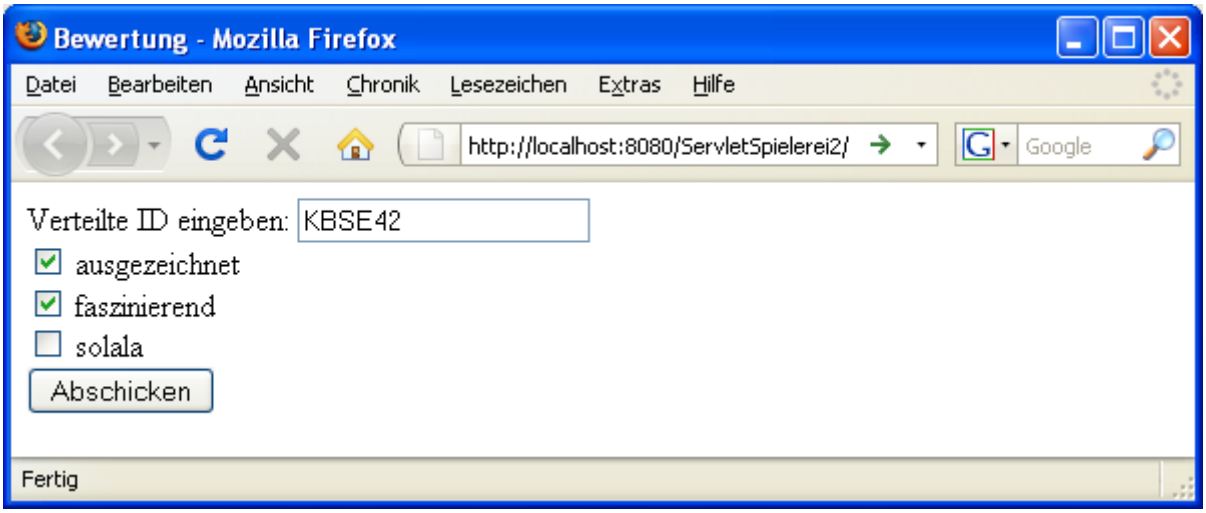
Die resultierende web.xml-Datei sieht wie folgt aus, die Startseite kann auch direkt als „welcome-file“ in der XML-Datei eingetragen werden.

web.xml *

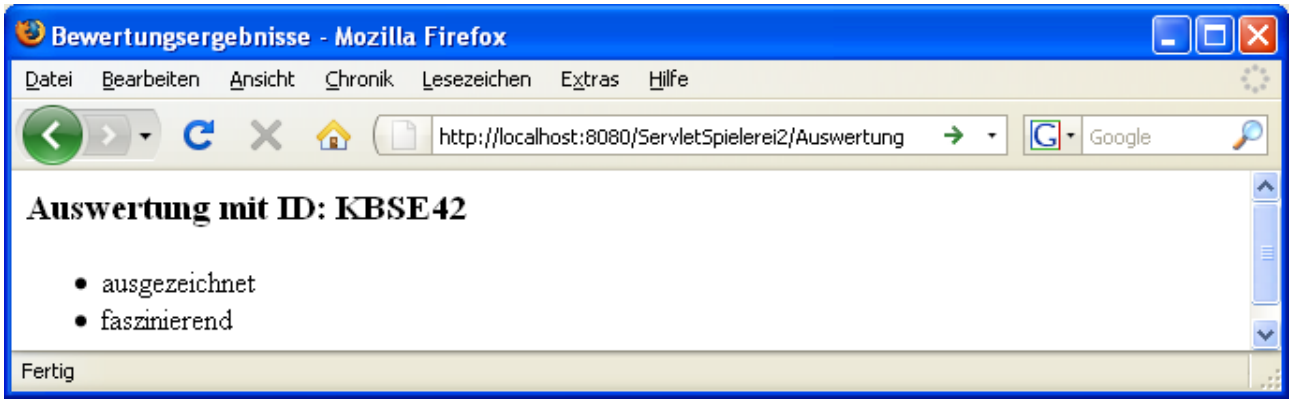
General | Servlets | Filters | Pages | References | Security | XML

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.5"
3      xmlns="http://java.sun.com/xml/ns/javaee"
4      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
5       <servlet>
6           <servlet-name>Auswertung</servlet-name>
7           <servlet-class>servlets.Auswertung</servlet-class>
8           <load-on-startup>1</load-on-startup>
9           </servlet>
10      <servlet-mapping>
11          <servlet-name>Auswertung</servlet-name>
12          <url-pattern>/Auswertung</url-pattern>
13      </servlet-mapping>
14      <session-config>
15          <session-timeout>
16              30
17          </session-timeout>
18      </session-config>
19      <welcome-file-list>
20          <welcome-file>bewertung.html</welcome-file>
21          </welcome-file-list>
22      </web-app>
```

Wird dann das Projekt über „Run“ gestartet, öffnet sich am Ende der Default-Browser mit folgender Ausgabe, bei der dann „Abschicken“ gedrückt wird.



Man erhält dann folgendes Ergebnis.


Bewertungsergebnisse - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://localhost:8080/ServletSpielerei2/Auswertung
Google
Auswertung mit ID: KBSE42
ausgezeichnet
faszinierend
Fertig

10 JSF (NetBeans 8)

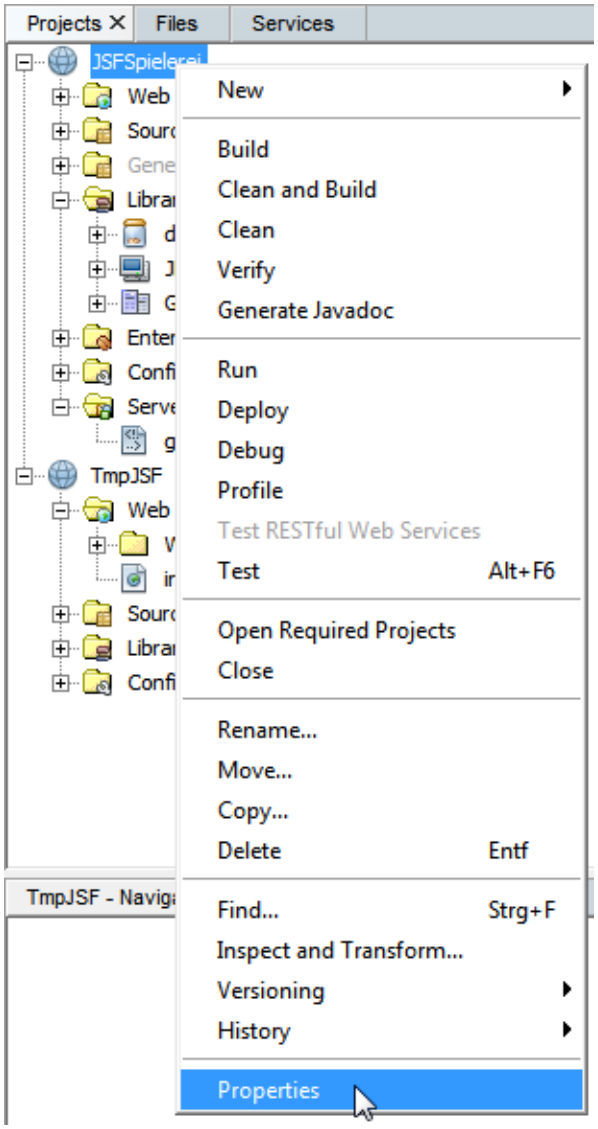
Im folgenden Beispiel wird die in NetBeans integrierte JSF-Version 2.2 benutzt. Für neuere Versionen oder Erweiterungen müssen diese im lib-Verzeichnis des jeweiligen Projekts zur Verfügung gestellt werden.

10.1 Anpassen der JDK-Version

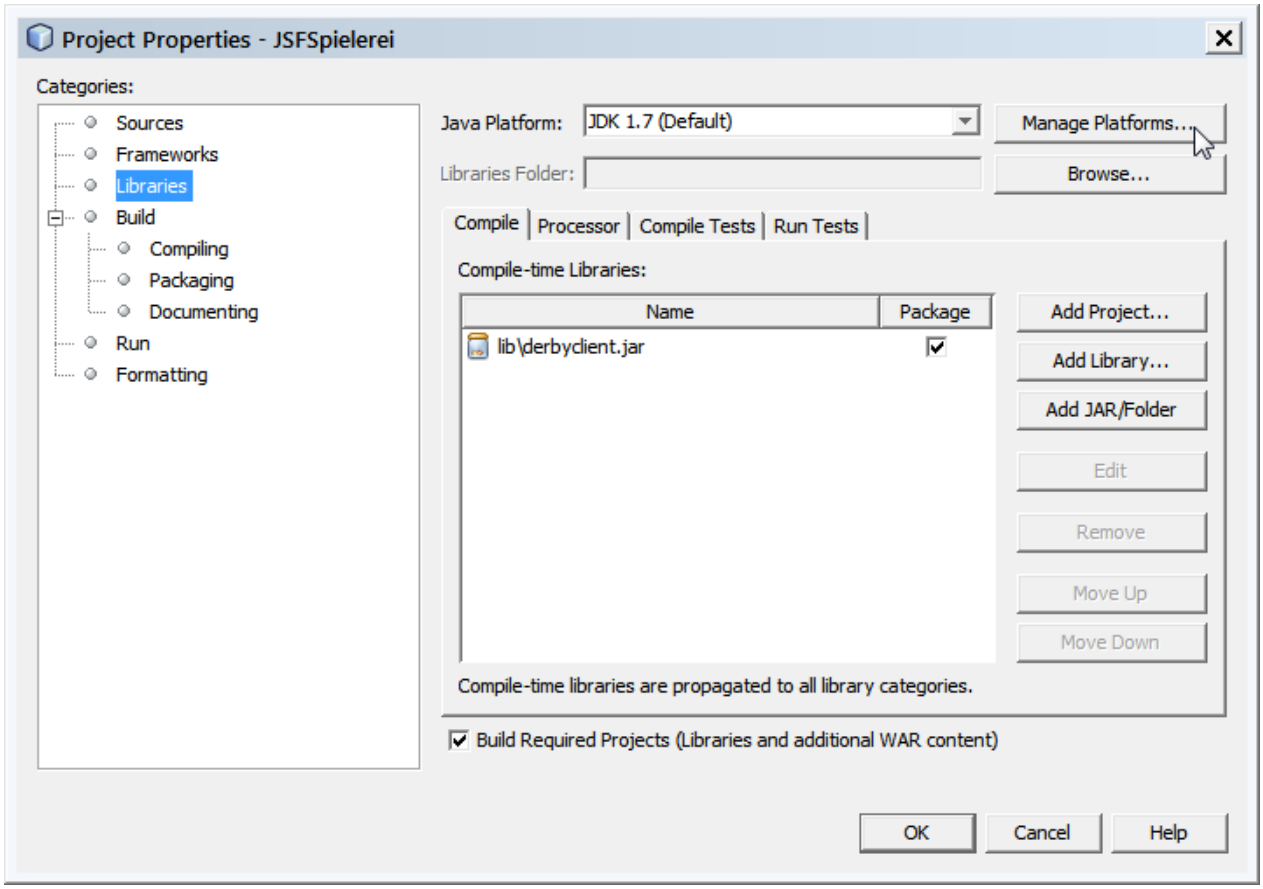
Dieses Unterkapitel *muss für NetBeans 8 nicht beachtet werden*, da JEE 7 sehr gut mit dem JDK 7 zusammen in NetBeans läuft. Personen, die sich nur für die aktuelle Version interessieren, können also direkt zum nächsten Teilkapitel springen. Generell bleiben die dokumentierten Schritte immer dann interessant, wenn eine bestimmte JDK-Version von mehreren auf dem Rechner befindlichen ausgewählt werden soll.

Zum Zeitpunkt der Erstellung dieses Abschnitts liefen die meisten Java EE-Anwendungen und Frameworks nur zuverlässig mit einem JDK 6 und nicht JDK 7. Aus diesem Grund muss es eine JDK 6-Version auf dem Rechner geben. Das ist generell unproblematisch da mehrere Java-Version, auch des gleichen JDKs, dann mit unterschiedlicher Update-Nummer, problemlos nebeneinander installiert werden können.

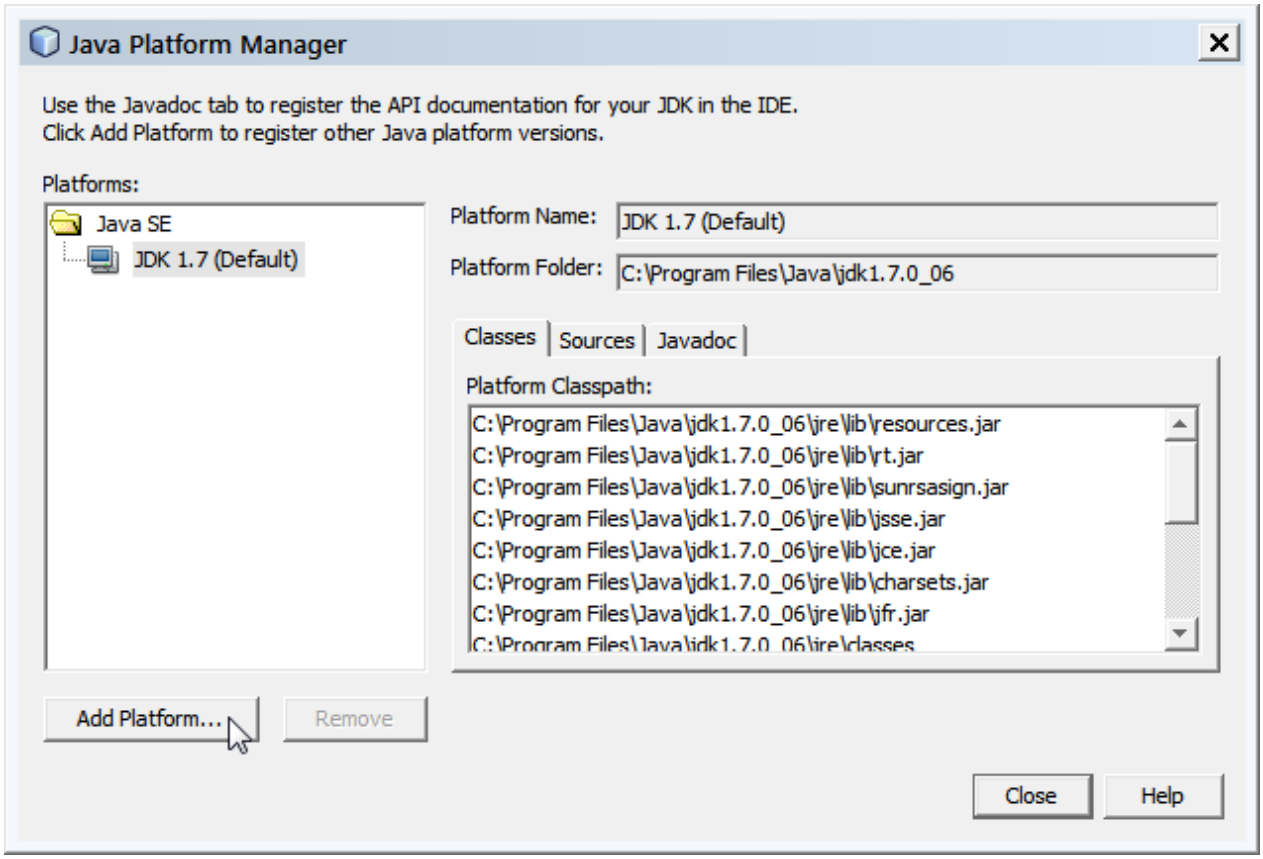
Die JDK 6-Nutzung wird wie folgt ermöglicht. Zunächst wird ein Rechtsklick auf einem Projekt gemacht und dann Properties gewählt.



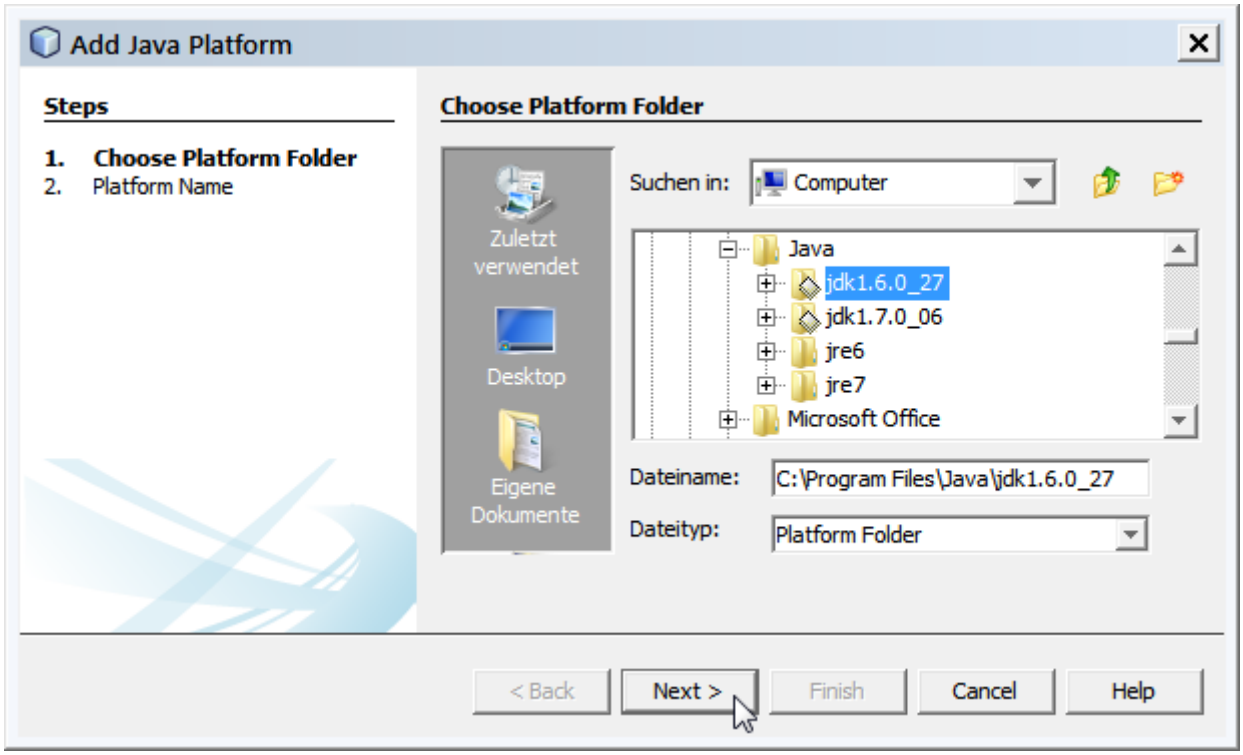
Links wird „Libraries“ geklickt und dann kann unter „Java Platform“ die Java-Variante für das Projekt gewählt werden. Ist noch kein JDK 1.6 in NetBeans bekannt, aber auf dem Rechner installiert, klickt man auf „Manage Platforms…“.



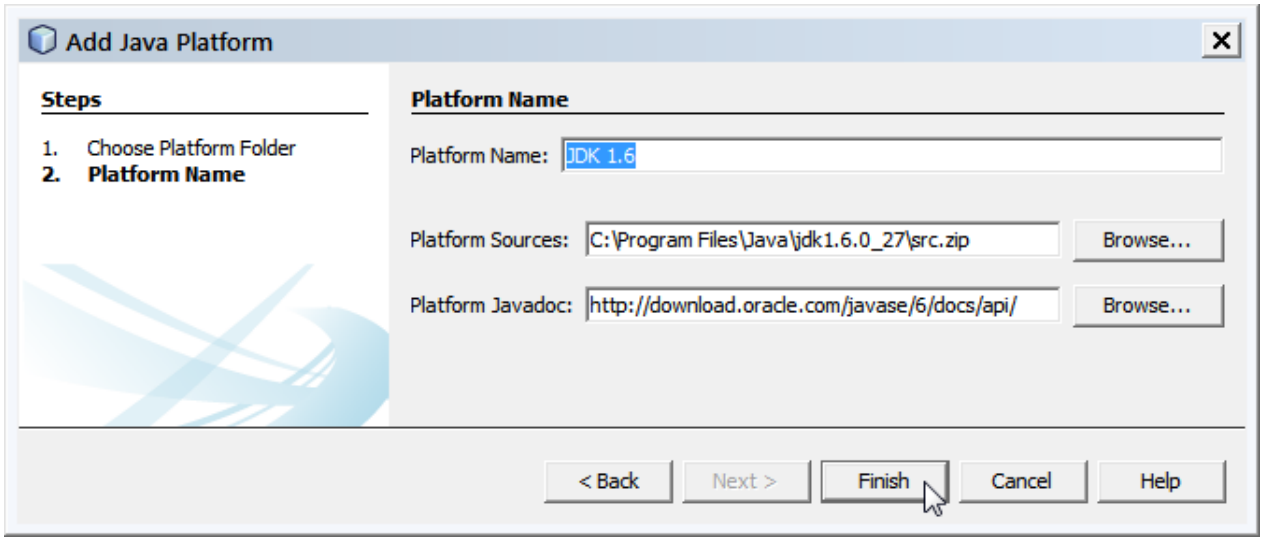
Hier klickt man links-unten auf „Add Plattform…“.



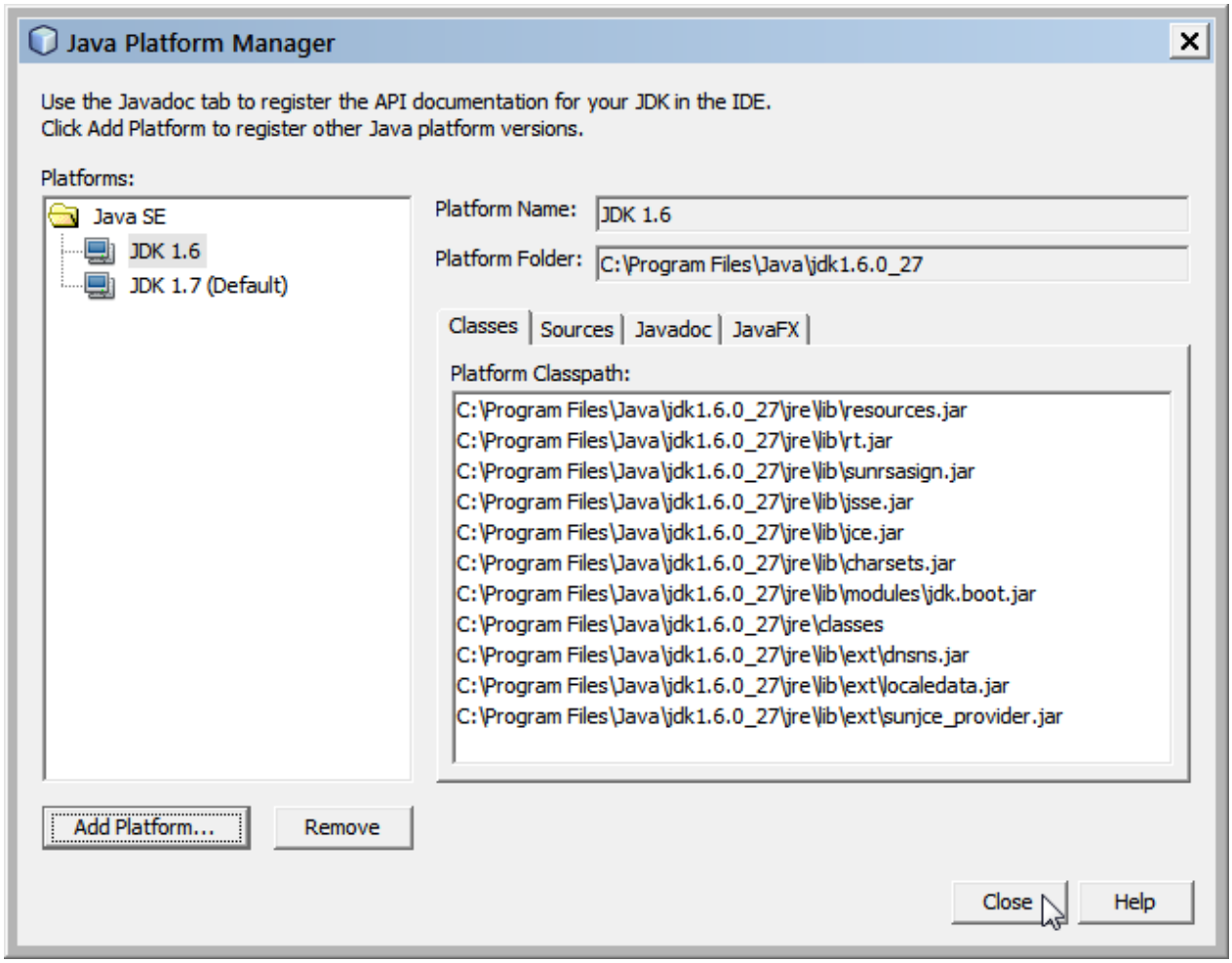
Man manöviert zum Installationsverzeichnis und klickt „Next>“.



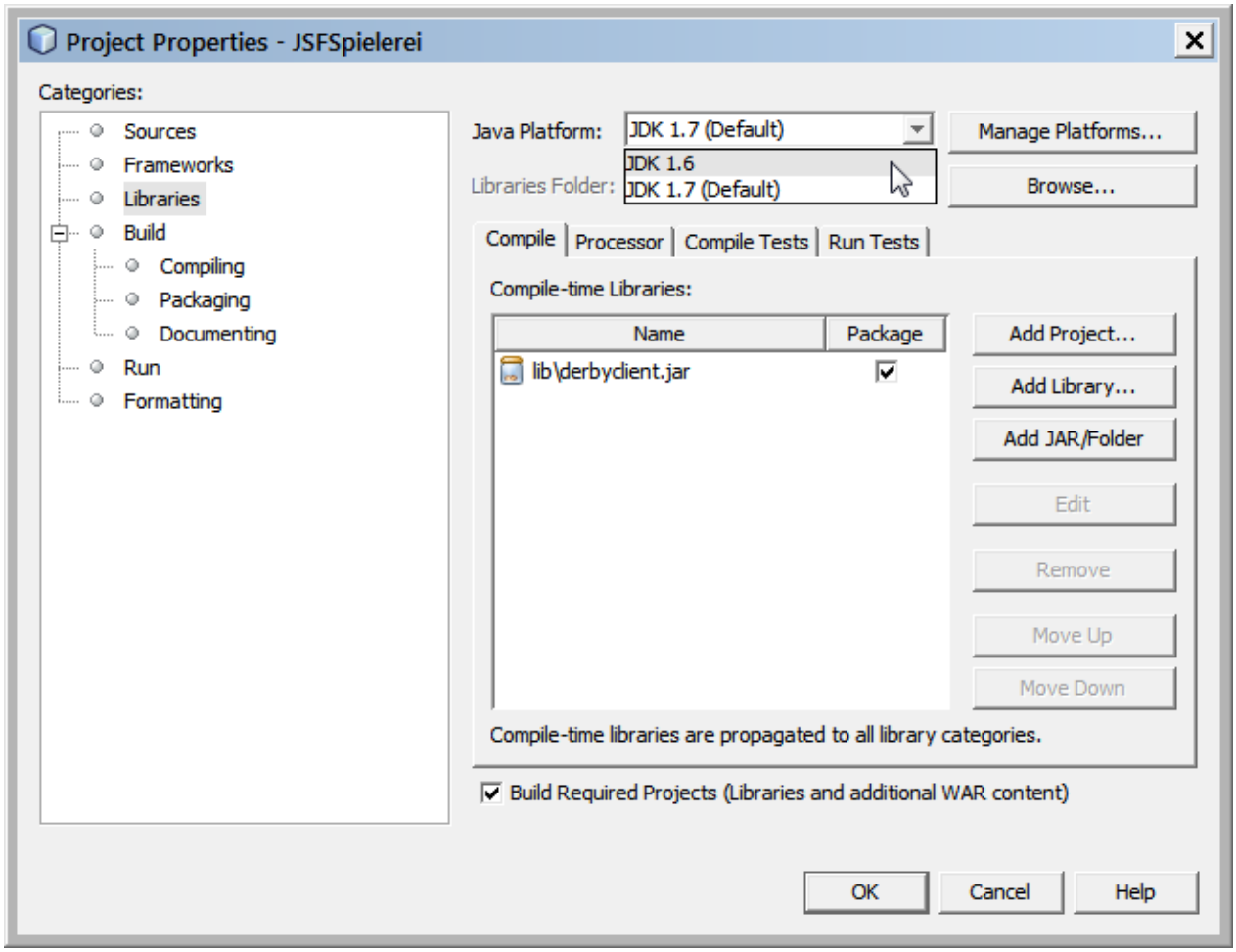
Die automatisch gefundenen Einstellungen werden mit „Finish“ übernommen.



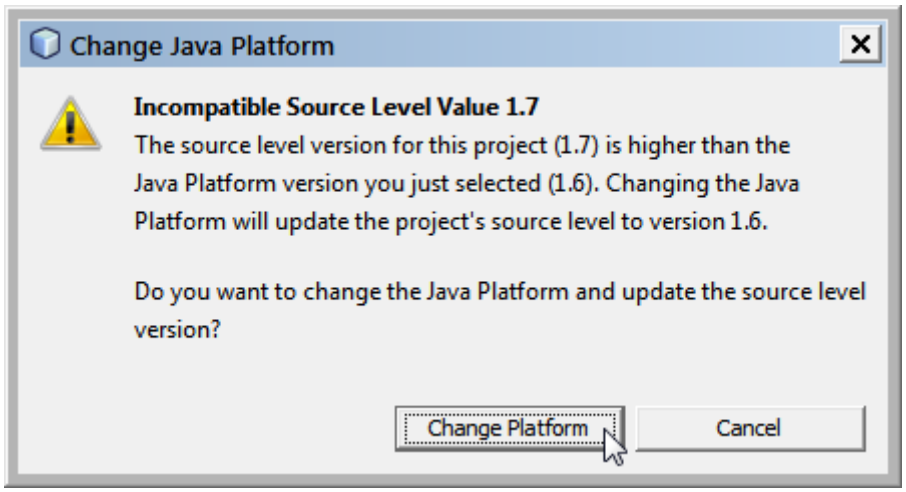
Die Integration in NetBeans wird mit „Close“ abgeschlossen.



Nun kann das JDK für dieses Projekt und danach auch einfach für andere Projekte genutzt werden.

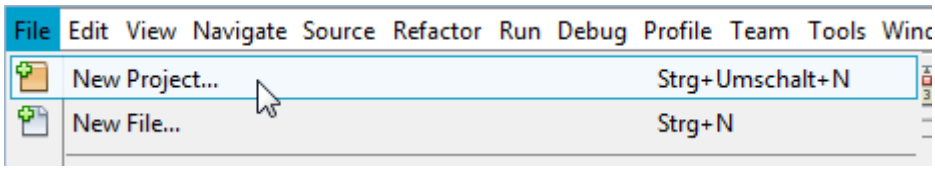


Eine Meldung der folgenden Art ist zu bestätigen.

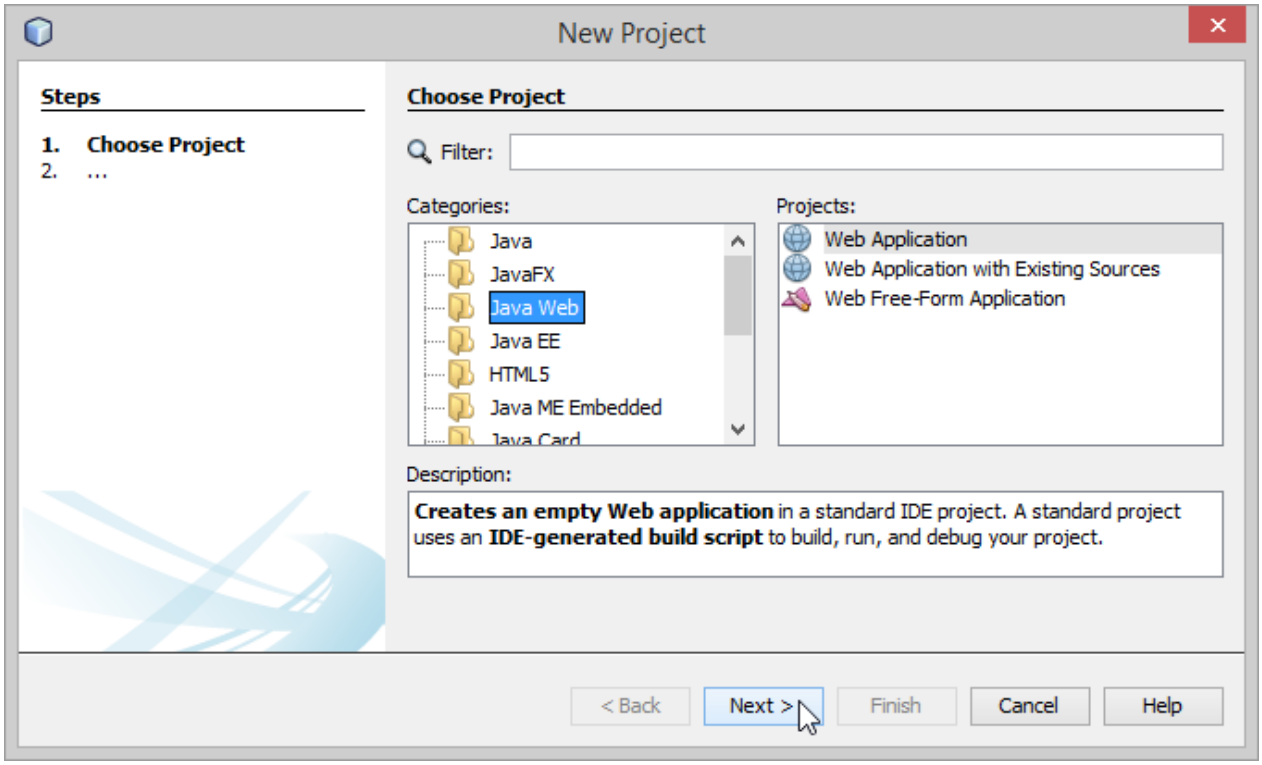


10.2 Erstellung eines JSF-Projekts

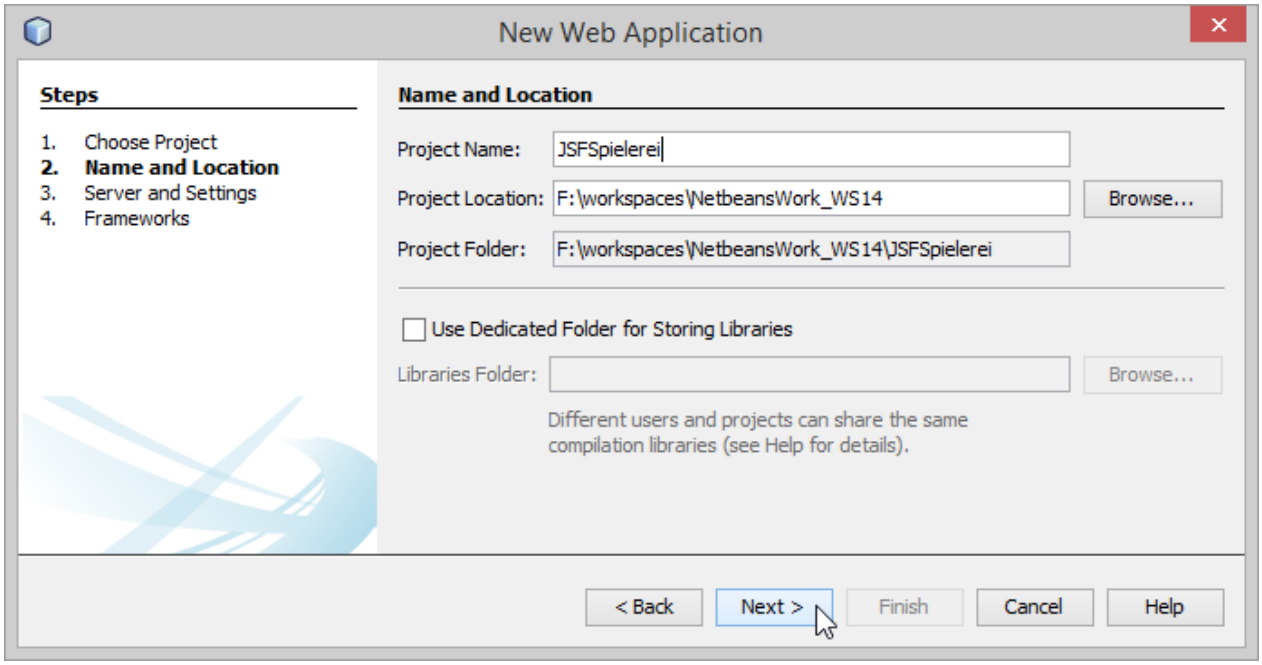
Die Projekterstellung beginnt z. B. über „File“ und „New Project...“.



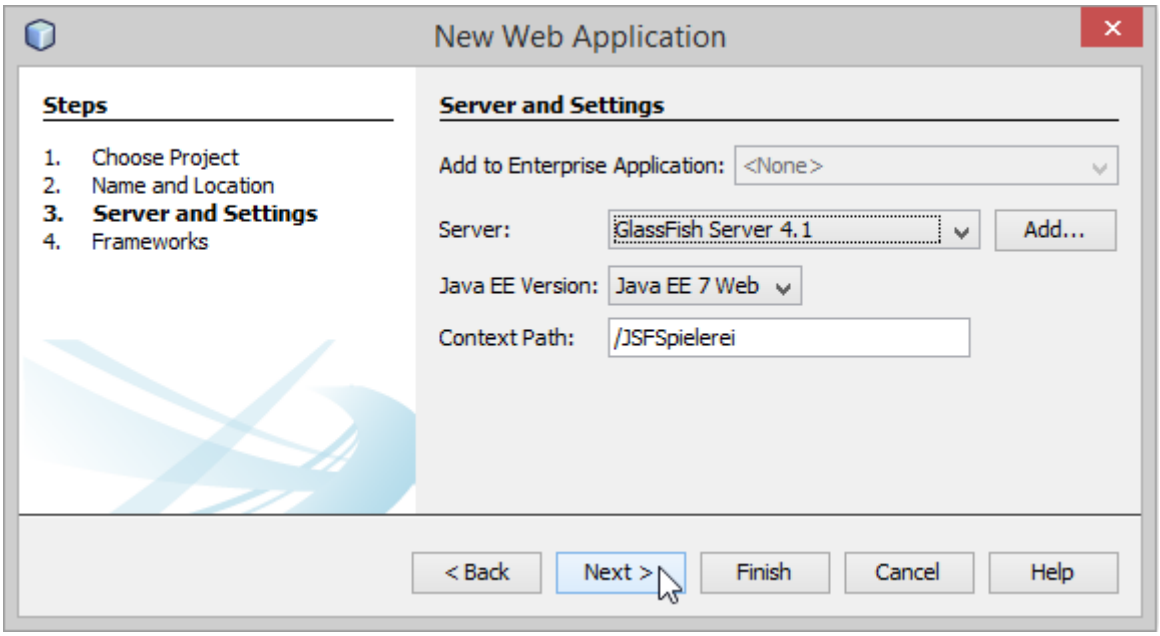
Hier wird unter der Kategorie „Java Web“ dann „Web Application“, dann „Next>“ geklickt.



Nach der Vergabe eines Projektnamens kann „Next>“ gedrückt werden. Der vorgeschlagene Pfad sollte zu den „üblichen“ Projekten führen, könnte aber beliebig gewählt werden.

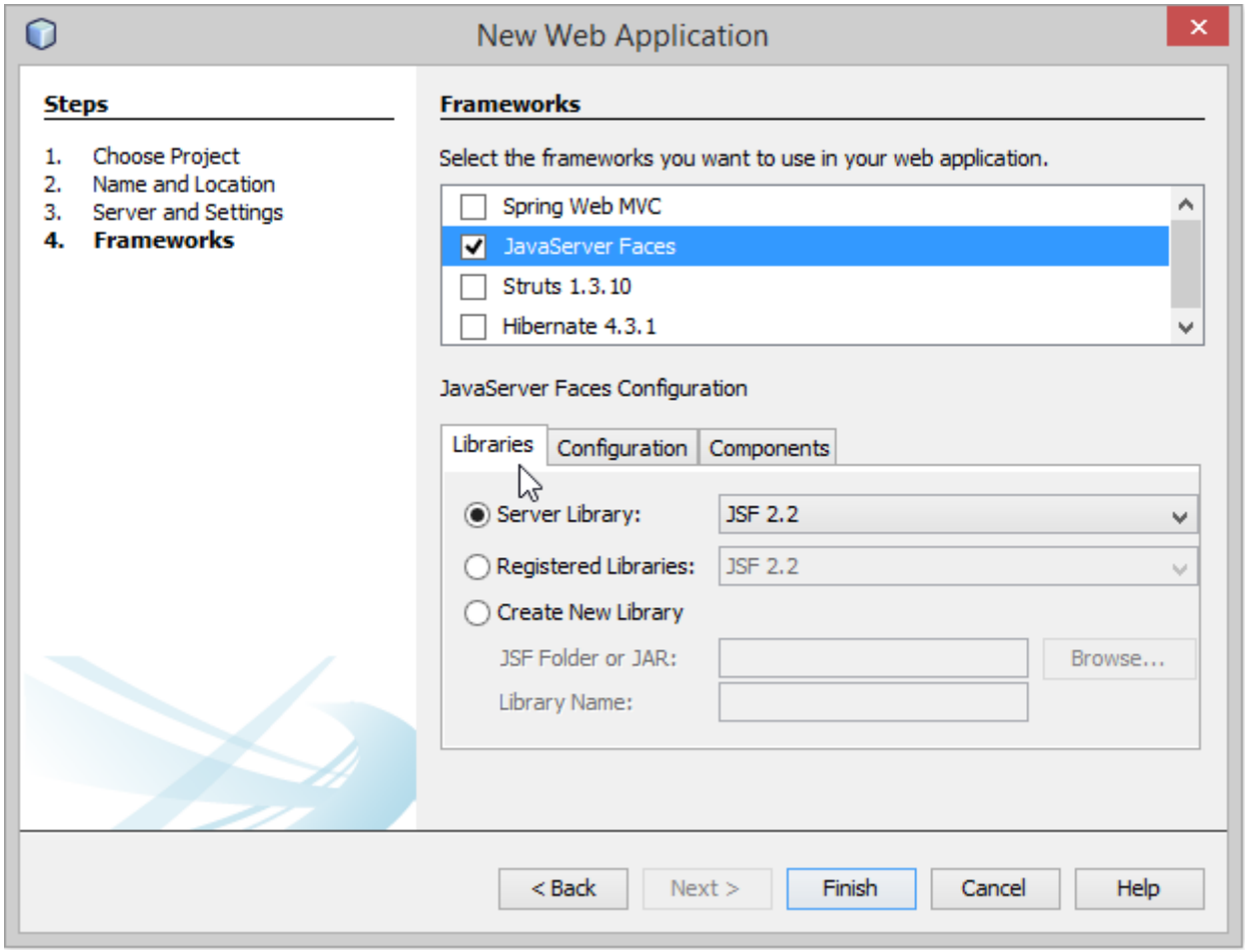


Die folgenden Einstellungen können übernommen werden, dabei wird davon ausgegangen, dass GlassFish 4.0 und ein JDK der Version 7 oder neuere Versionen auf dem Rechner lokal installiert ist. Hier kann auch Apache Tomcat oder jeder andere installierte und in NetBeans bekannt gemachte Server (2.3 Einbindung eines neuen GlassFish-Servers in NetBeans) ausgewählt werden.

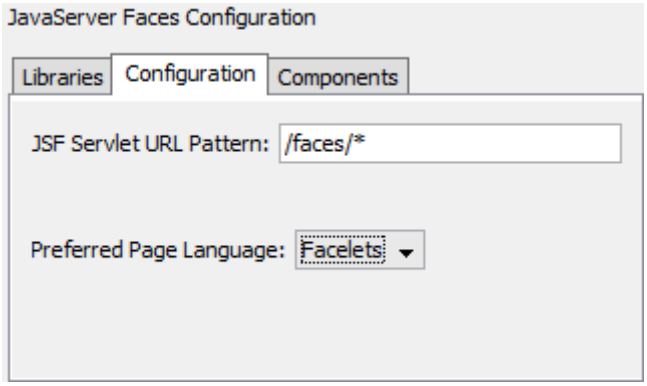


Jetzt wird die JSF-Erweiterung mit einem Kreuz bei „JavaServer Faces“ ausgewählt, worauf unten weitere Detailinformationen sichtbar werden. Beim Klick in das Auswahlfeld kann es etwas dauern,

bis der Haken angezeigt wird. Die weiteren Frameworks sind auch später bei Bedarf für das Projekt ergänzbar.

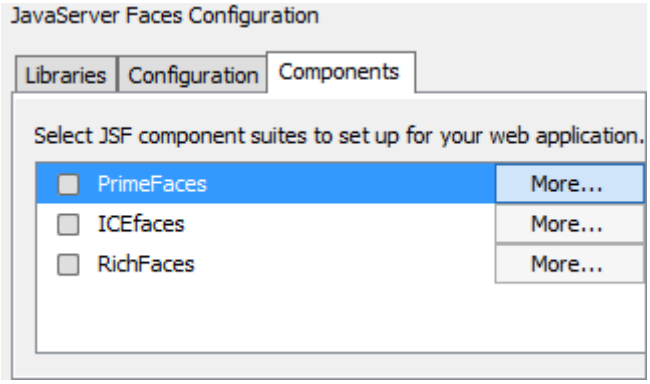


Auch die Einstellungen unter dem Reiter „Configuration“ können so übernommen werden.

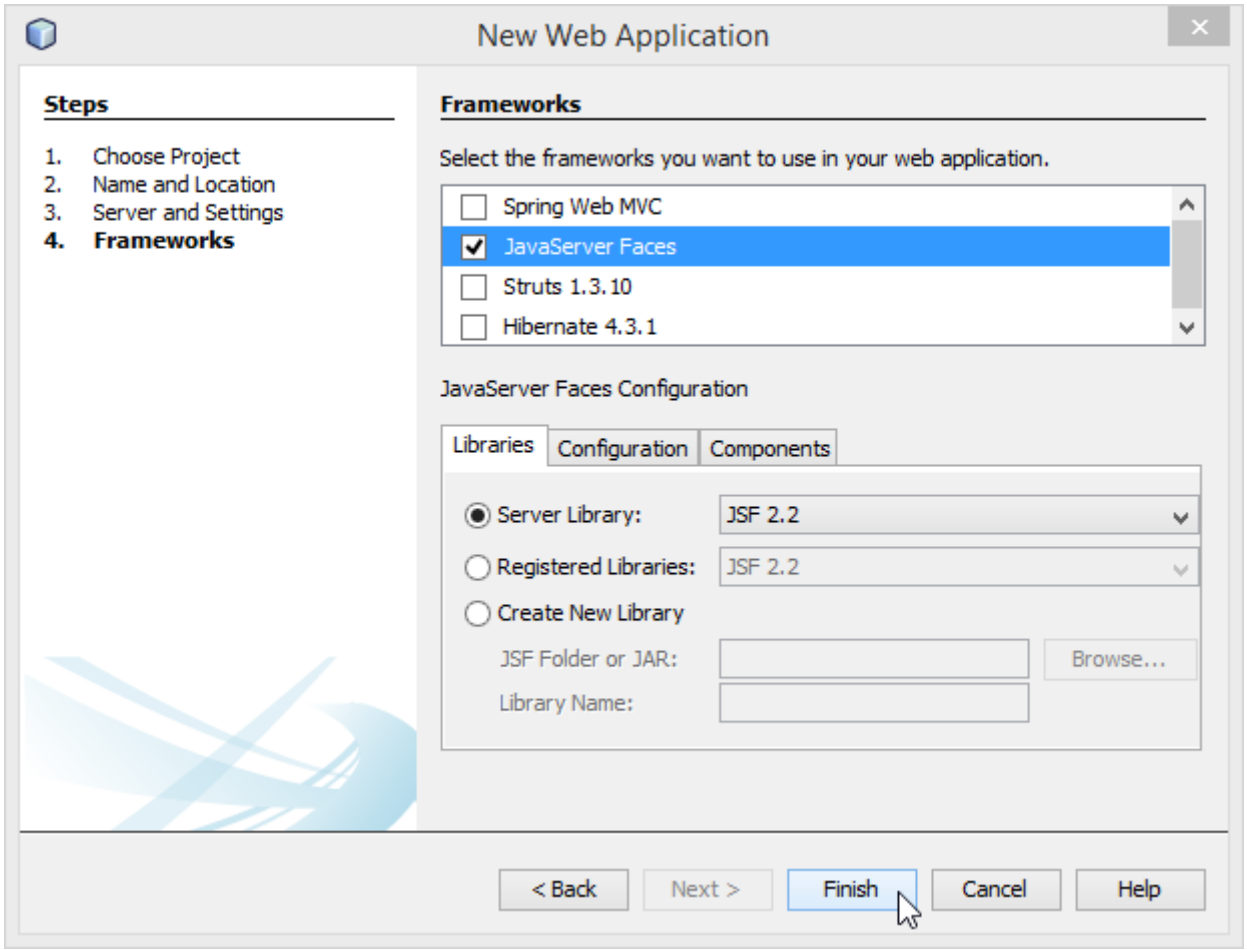


Unter „Components“ kann man bereits einige JSF-Erweiterungen auswählen, die alternativ aber auch direkt über Jar-Dateien in das Projekt eingebunden werden können. Hier wird keine Auswahl

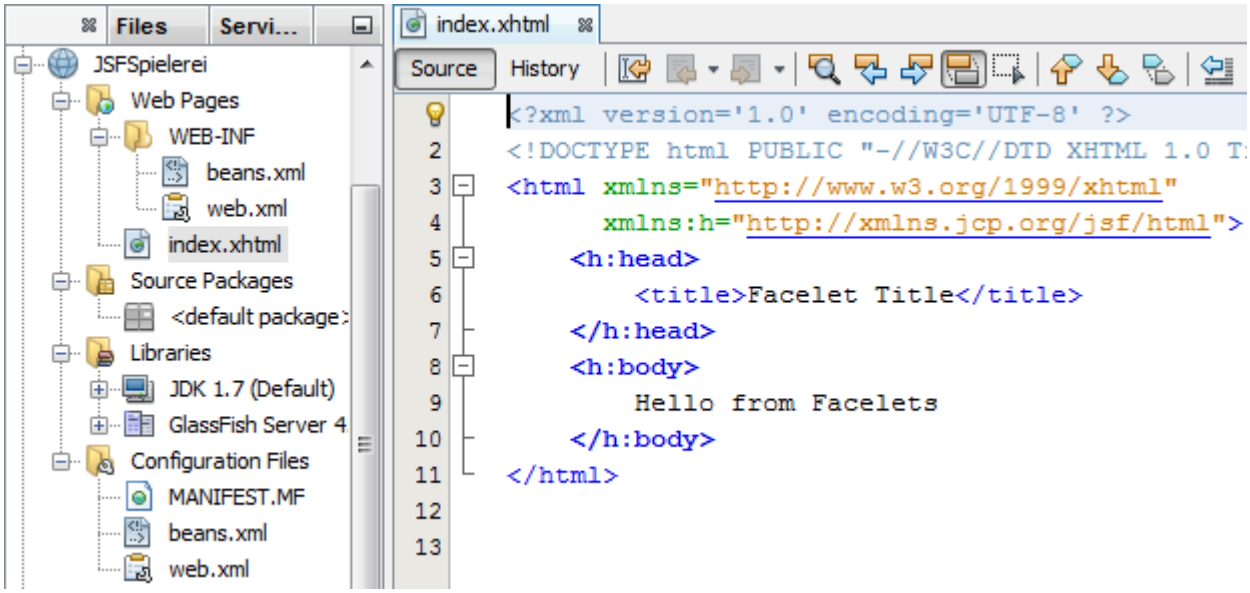
getroffen.



Die JSF-Projekterzeugung wird mit „Finish“ abgeschlossen.



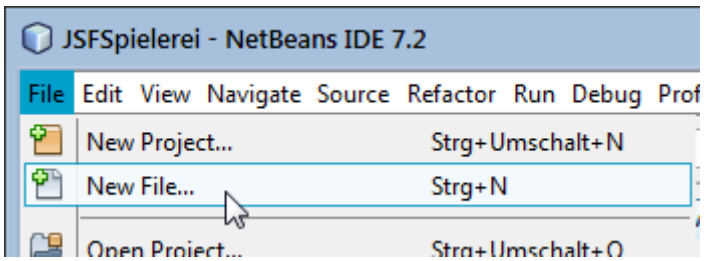
Das resultierende Projekt hat die folgende Form, es wurde noch keine faces-config.xml angelegt. Die bereits vollständig lauffähige Seite index.xhtml zeigt die Einbindung einer der JSF-Bibliotheken (Libraries).



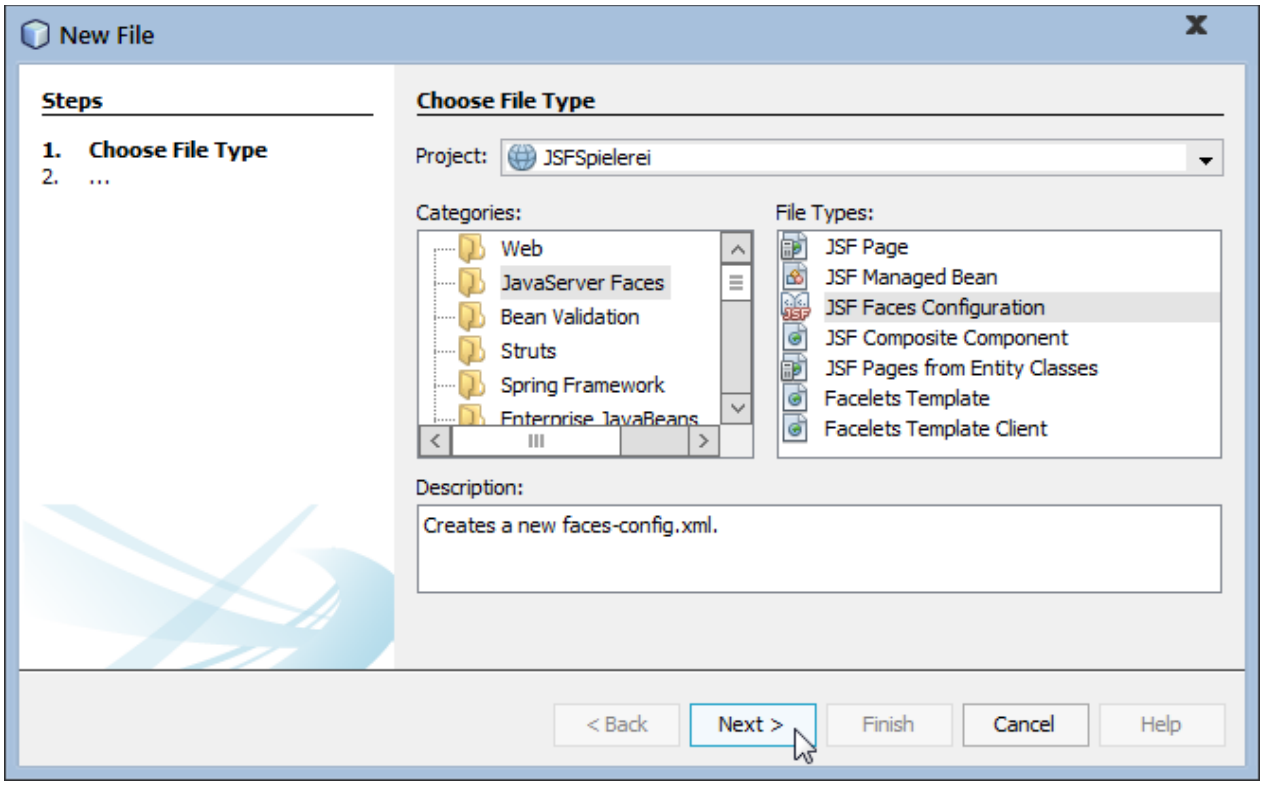
In der Datei web.xml stehen wichtige Projekteinstellungen, die vom Server beim Einspielen der Applikation genutzt werden. Neben der Zuordnung der Zugriffe auf diese Applikation zur JSF-Bearbeitung im unteren Teil, steht in der im Folgenden gezeigten Datei auch der Status des Projekts mit Development oder Production. Beim Status Development werden alle erkannten Fehler automatisch unten auf der als nächstes angezeigten Seite eingeblendet, bei PRODUCTION werden nur explizit vom Nutzer geforderte Ausgaben erzeugt.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

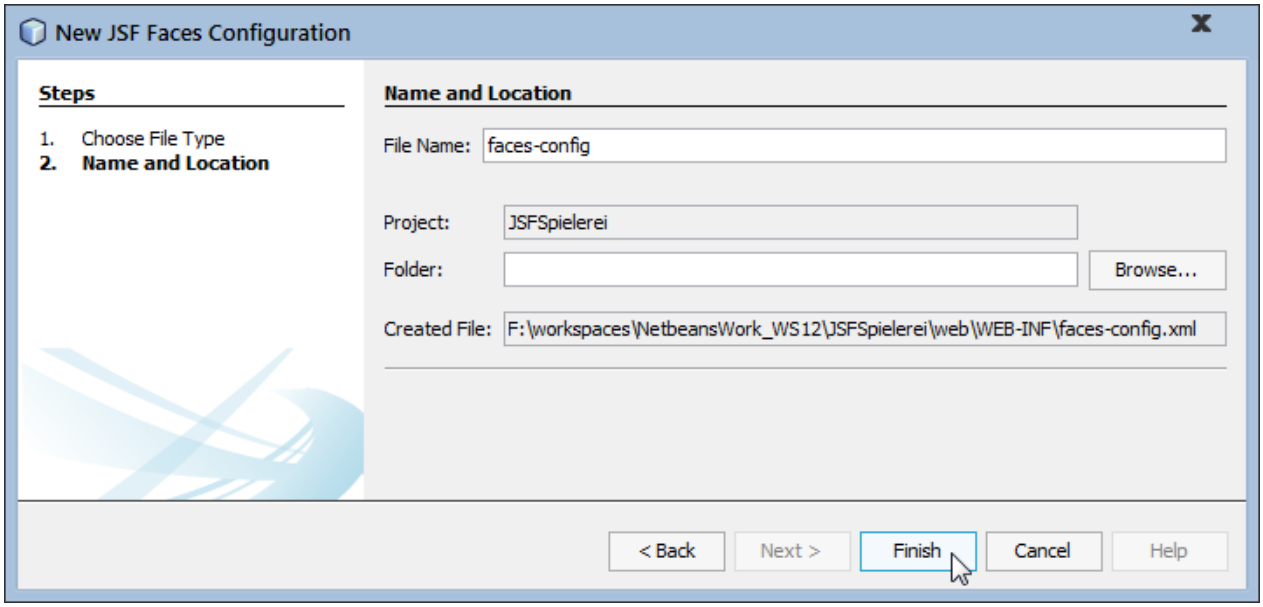
Formal kann man ab JSF 2.0 versuchen, ohne die Konfigurationsdatei faces-config.xhtml auszukommen, generell ist sie aber für viele Einstellungen nützlich, deshalb ist es manchmal sinnvoll, diese Datei mit dem Anlegen des Projektes zu erzeugen. Dazu wird unter „File“ der Punkt „New File...“ gewählt.



Dann wählt man auf der linken Seite „JavaServer Faces“ und rechts dann “JSF Faces Configuration“.



Auf der folgenden Seite kann einfach „Finish“ geklickt werden.

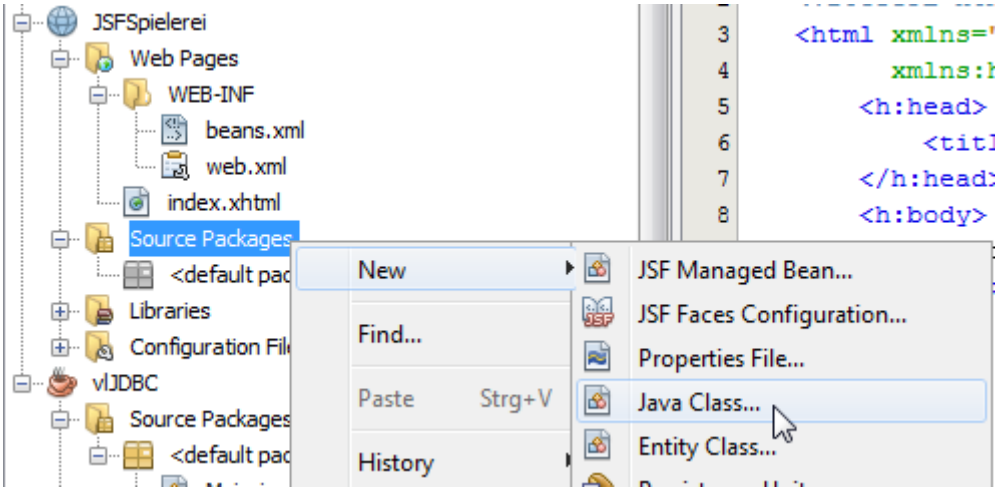


Hinweis: In den folgenden Beschreibungen wird davon ausgegangen, dass keine faces-config.xml angelegt wurde, wobei darauf an der einen relevanten Stelle eingegangen wird.

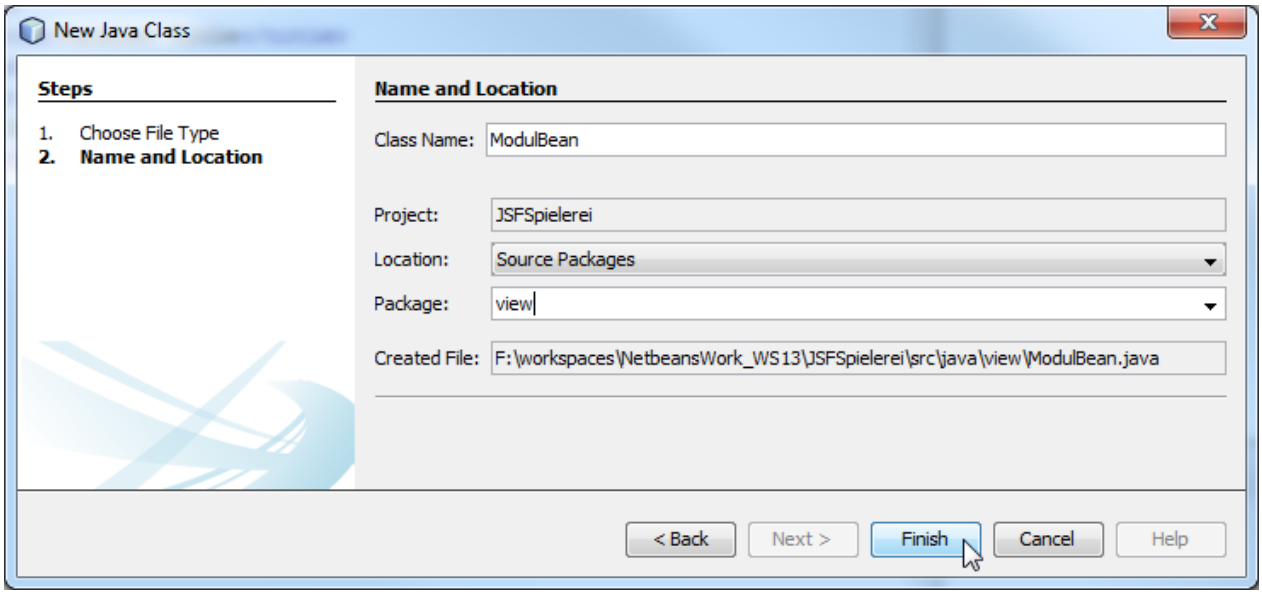
10.3 Entwicklung eines Minimalprojekts

Das folgende Beispiel zeigt, wie man ein kleines Projekt mit JSF und der Unterstützung von NetBeans entwickeln kann, dabei wird nur etwas Basisfunktionalität gezeigt und kein Wert auf eine erweiterbare Softwarearchitektur z. B. mit Controller-Klassen gelegt. Ausgangspunkt ist ein leeres Projekt, wie es im vorherigen Abschnitt angelegt wurde. Es soll ein Eingabefenster zur Eingabe von Modul-Informationen (Name des Moduls, Nummer) geben, die nach einem Knopfdruck in einer Ausgabeseite angezeigt werden. Benötigt werden dazu zwei Web-Seiten, zur Eingabe index.xhtml und zur Ausgabe ausgabe.xhtml. Die Namen sind willkürlich und können, passend zu projektindividuellen Codierungs-Standards, angepasst werden.

Die Steuerungs-Klasse ModulBean kann wie eine normale Java-Klasse angelegt werden. Hier wird davon ausgegangen, dass die Sichtbarkeiten nicht in der faces-config.xml festgelegt werden sollen, sondern in Annotationen stehen. Die Erstellung beginnt z. B. durch einen Rechtsklick auf „Source Packages“ und der Auswahl „New“ und „Java Class“. Falls „Java Class“ nicht direkt sichtbar ist, muss dies unter „Other“ gesucht werden.



Es wird der Klassenname (Class Name) und ein Paket (Package) eingetragen und „Finish“ geklickt.



Das Ergebnis sieht zunächst wie folgt aus.

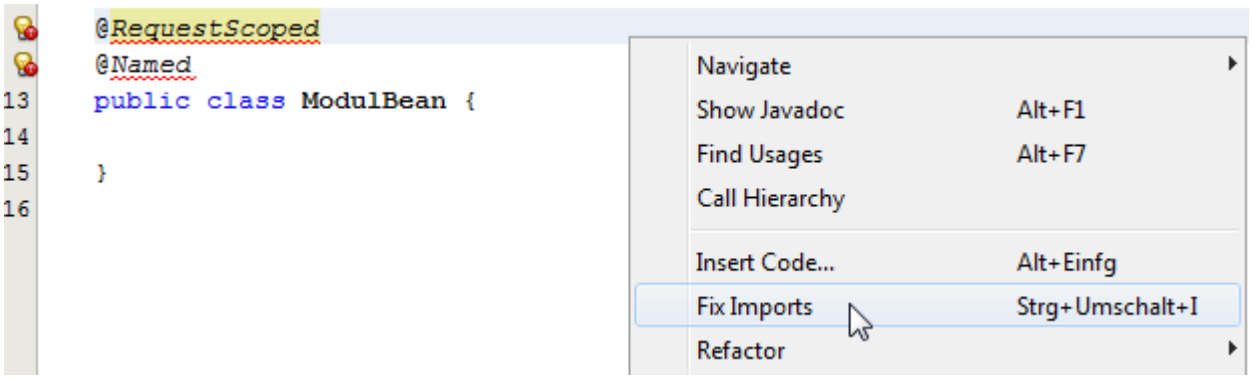
```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package controller;

/**
 *
 * @author X
 */
public class ModulBean {
```

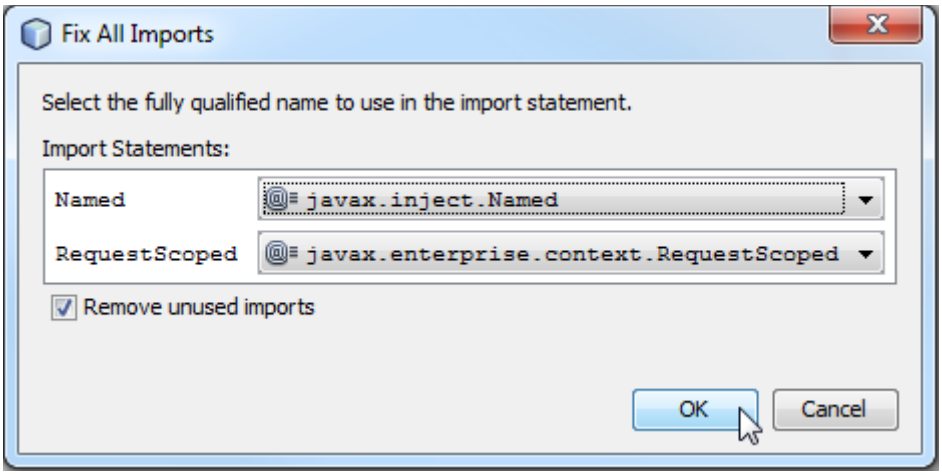
```
}
```

Nun werden zwei Annotationen ergänzt, die eine macht Objekte der Klasse in JSF-Seiten zugängig, die andere legt wieder den Sichtbarkeitsbereich fest. Man erkennt, dass die Annotationen zunächst nicht bekannt sind. Da die Annotationen aus CDI stammen, muss zunächst eine dazu passende Bibliothek eingebunden werden, die GlassFish bereits mitbringt. Dazu wird ein Rechtsklick auf das Verzeichnis „Libraries“ gemacht und „Add Library“ ausgewählt.

Nun kann z. B. mit einem Rechtklick irgendwo im Code-Fenster gemacht und „Fix Imports“ gewählt werden.



Ein zentraler Fehler wäre es, hier immer den ersten Auswahlen zu vertrauen. Es ist zu beachten, dass die Klassen javax.inject.Named und javax.enterprise.context.RequestScoped benötigt werden.



Es wird davon ausgegangen, dass die Sichtbarkeit durch Annotationen festgelegt wird.

Es werden folgende Methoden ergänzt, wobei die Entwicklung immer in Iterationen stattfindet, in denen die XHTML-Seiten parallel zur eigentlichen Software-Entwicklung entstehen. Hat das Projekt irgendwann eine gewisse Stabilität erreicht, können die Entwicklungsaufgaben getrennt angegangen werden. Wichtig ist nur die frühzeitige Klärung der Schnittstelle zwischen Oberfläche (XHTML) und

der Geschäftslogik in Java. Die Umbenennung mit dem name-Attribut ist nicht notwendig, aber nicht verboten.

```
package controller;

import java.io.Serializable;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;

@RequestScoped
@Named(value = "modul")
public class ModulBean implements Serializable {
  private static final long serialVersionUID = 1L;
  private int nr;
  private String name;

  public ModulBean(){}

  public ModulBean(int nr, String name) {
    this.nr = nr;
    this.name = name;
  }

  public String uebernehmen(){
    return "ausgabe.xhtml";
  }

  public String eingeben(){
    return "index.xhtml";
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public int getNr() {
    return nr;
  }

  public void setNr(int nr) {
    this.nr = nr;
  }

  @Override // generieren lassen
  public boolean equals(Object object) {
    if (object==null || !(object instanceof ModulBean)) {
      return false;
    }
```

```
    ModulBean other = (ModulBean) object;
    if (this.nr != other.nr  || !this.name.equals(other.name)) {
      return false;
    }
    return true;
  }

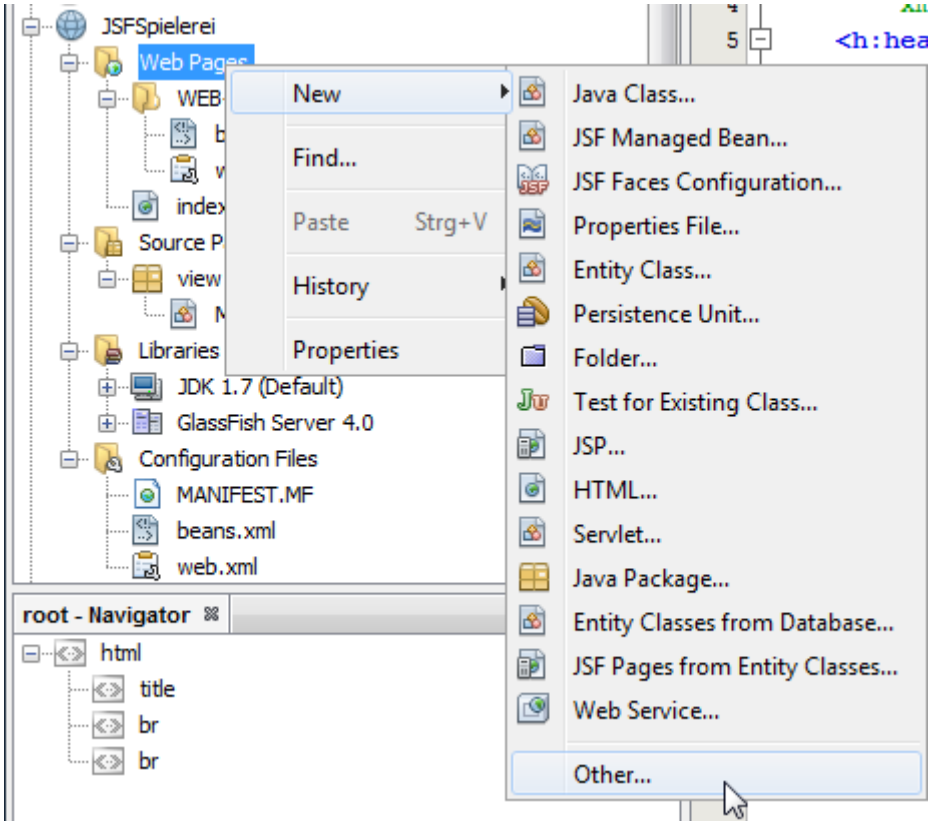
  @Override // generieren lassen
  public int hashCode() {
    int hash = 5;
    hash = 47 * hash + this.nr;
    hash = 47 * hash + (this.name != null ? this.name.hashCode() : 0);
    return hash;
  }

  @Override
  public String toString() {
    return name+"("+nr+")";
  }
}
```

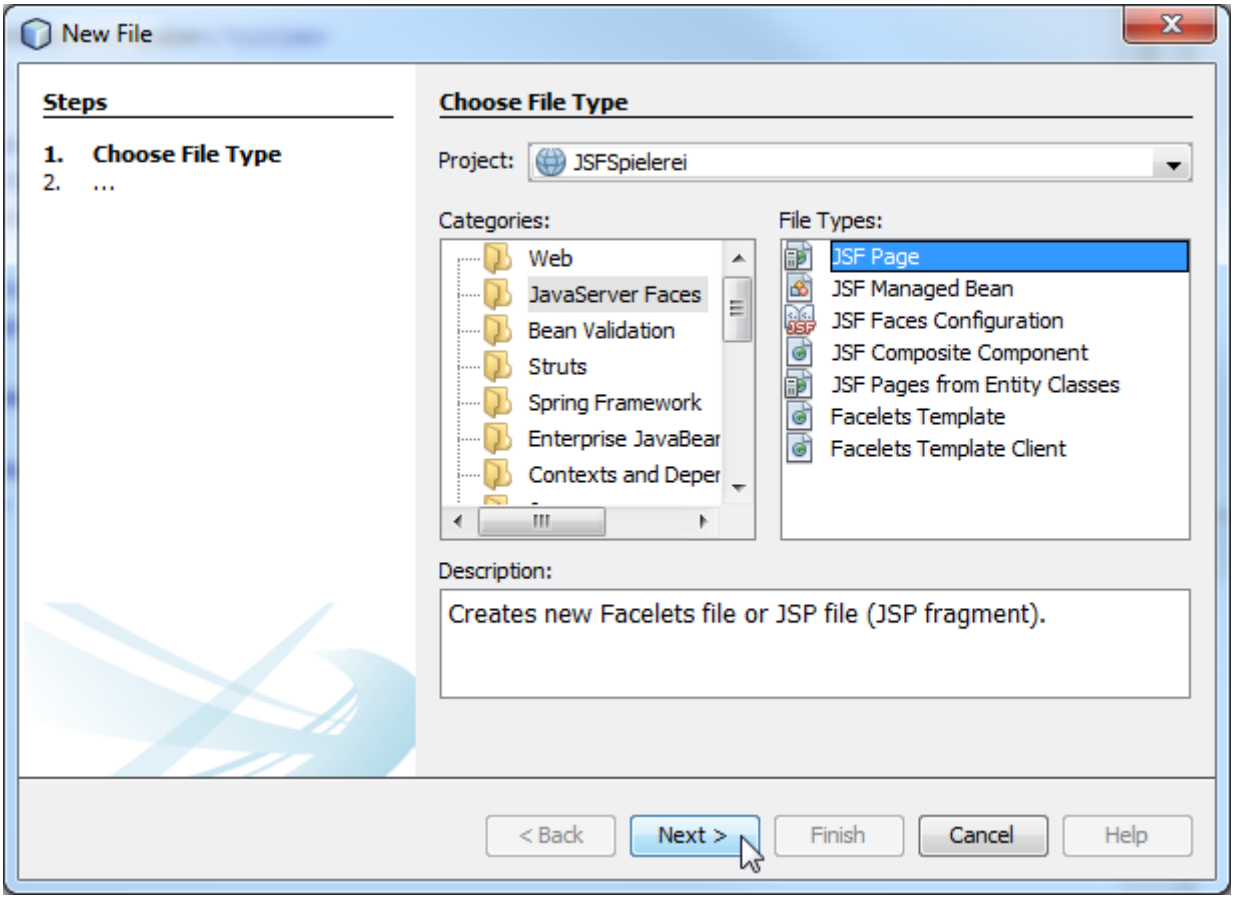
Die Startseite index.xhtml der Applikation hat dabei folgenden Inhalt.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Moduleingabe</title>
  </h:head>
  <h:body>
    <h:form>
      <h:outputText value="Modulname "/>
      <h:inputText id="mname" value="#{modul.name}"
                   required="true"/><br/>
      <h:outputText value="Modulnummer "/>
      <h:inputText id="mnr" value="#{modul.nr}"
                   required="true"/><br/>
      <h:commandButton value="Abschicken"
                       action="#{modul.uebernehmen}"/>
    </h:form>
  </h:body>
</html>
```

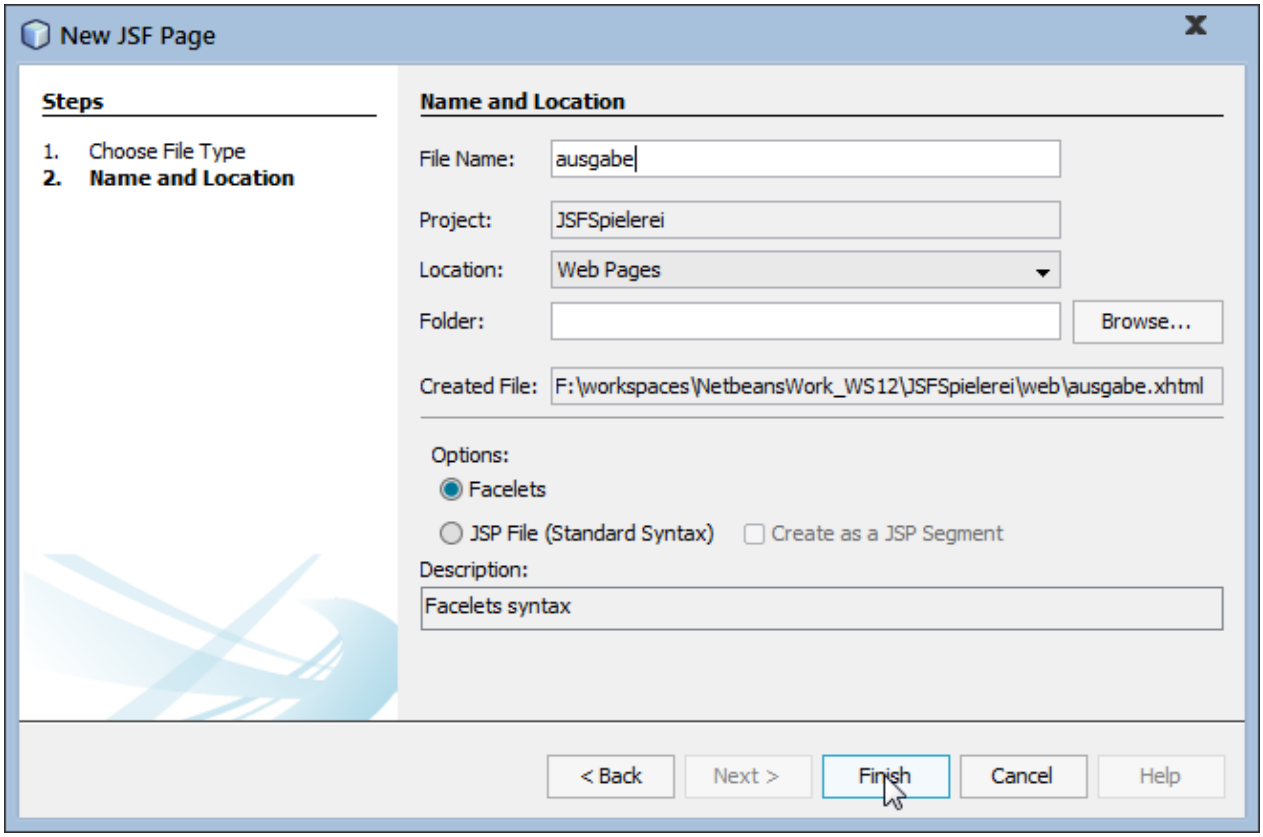
Die zweite Seite ausgabe.xhtml kann über viele Wege erstellt werden. Ein Weg ist die direkte Erzeugung im Projekt. Dazu wird ein Rechtsklick auf dem Ordner Web Pages gemacht, „New>“ und dann unten „Other...“ gewählt.



Dann wird auf der linken Seite „JavaServer Faces“ und rechts „JSF Page“ gewählt.



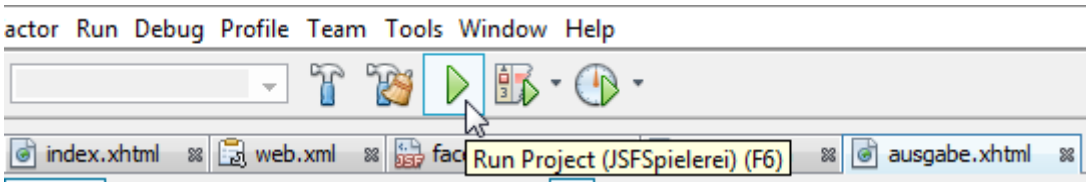
Nun muss nur noch der Name der neuen Datei neben „File Name:“ eingetragen werden. Die Dateierzeugung wird dann mit „Finish“ abgeschlossen.



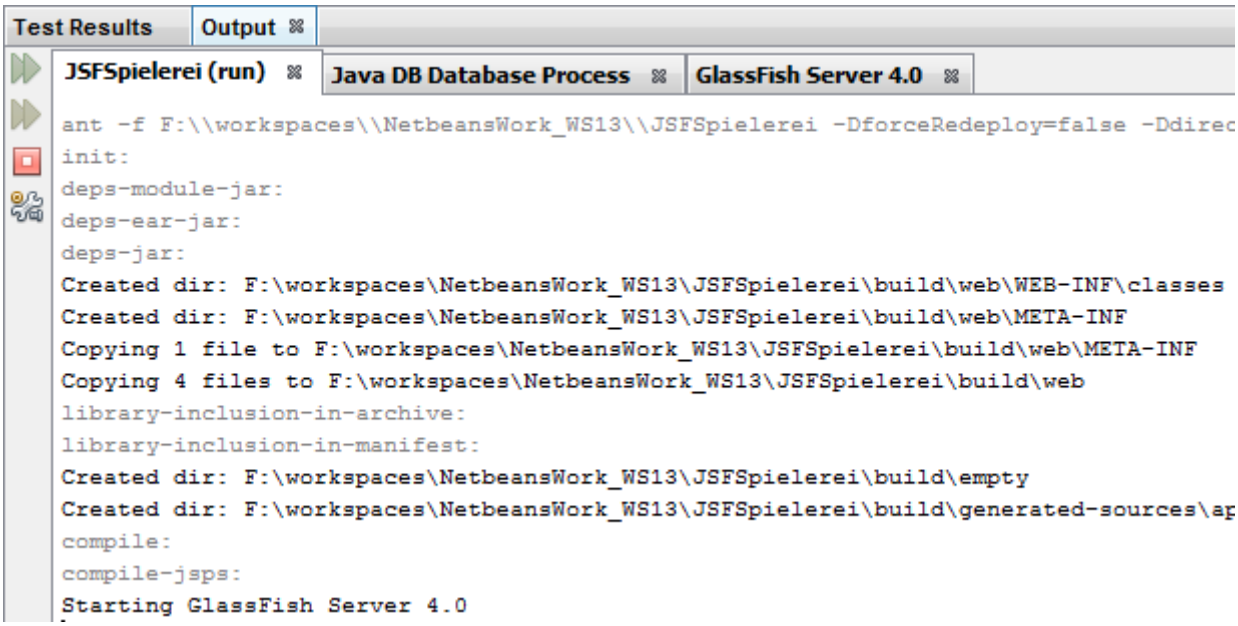
Letztendlich muss die Datei dann noch mit sinnvollem Inhalt gefüllt werden, der wie folgt aussehen kann.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Modulausgabe</title>
  </h:head>
  <h:body>
    <h:form>
      <h:outputText value="Modulname: "/>
      <h:outputText id="mname" value="#{modul.name}"/> <br/>
      <h:outputText value="Modulnummer: "/>
      <h:outputText id="mnr" value="#{modul.nr}"/><br/>
      <h:commandButton value="Zur Eingabe"
                       action="#{modul.eingeben}"/>
    </h:form>
  </h:body>
</html>
```

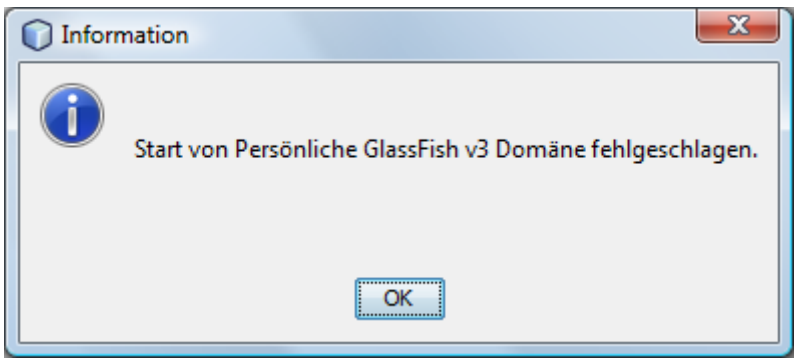
Die Applikation wird auf dem üblichen Weg gestartet. Generell sollte man bei Änderungen der Dateistrukturen oder bei neuen Dateien das Projekt vollständig neu übersetzen (Knopf links daneben mit „Clean and Build“).



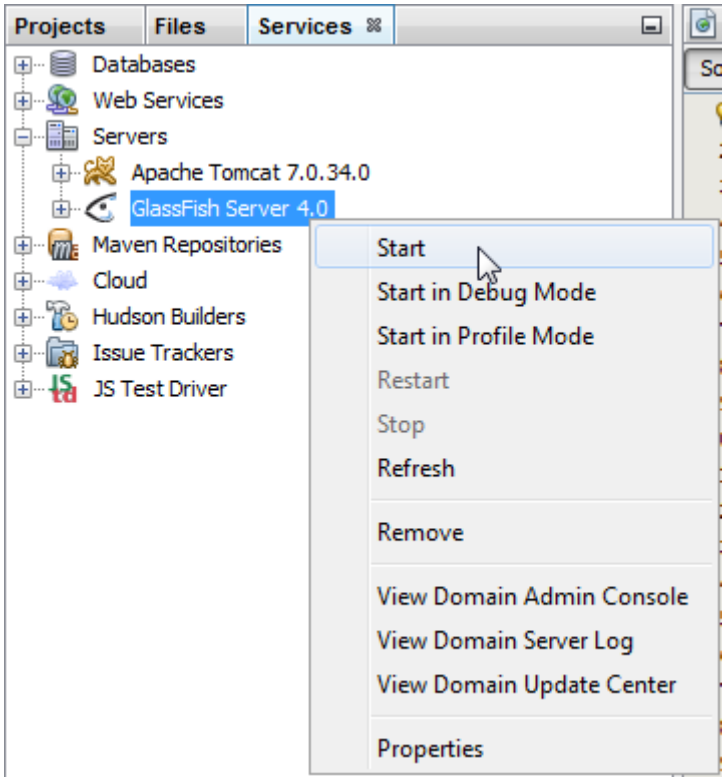
Nach der Übersetzung wird die eingepackte Applikation auf dem Web-Server automatisch installiert (deployed) und gestartet. Sollte der Server noch nicht laufen, wird er jetzt zuerst gestartet, was einige Zeit dauern kann.



Eventuell kann der Start zu einem internen Timeout mit folgender Fehlermeldung führen.



Den Fehler kann man durch den generell sinnvollen Schritt vermeiden, den Server bereits beim Start der Arbeiten selbst von Hand zu starten. Dies ist u. a. auch von NetBeans aus möglich. Dazu wird der obige Reiter „Dienste“ gewählt und die Einträge unter „Server“ aufgeklappt. Es können durchaus mehrere Server installiert sein, es sollte aber immer nur einer aus ernsthaften Performance-Gründen laufen. Durch einen Rechtsklick auf den Server kann man ihn dann im Menü starten.



Man kann am Servertext durch ein kleines Icon seinen Zustand erkennen, wobei kein Icon bedeutet, dass der Server nicht läuft.

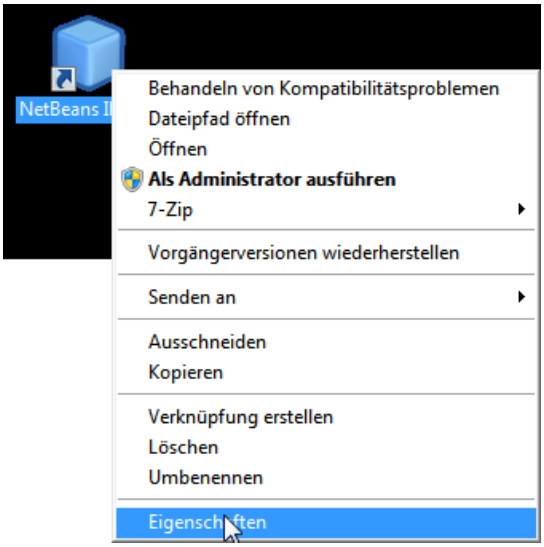
Eieruhr für startenden Server:



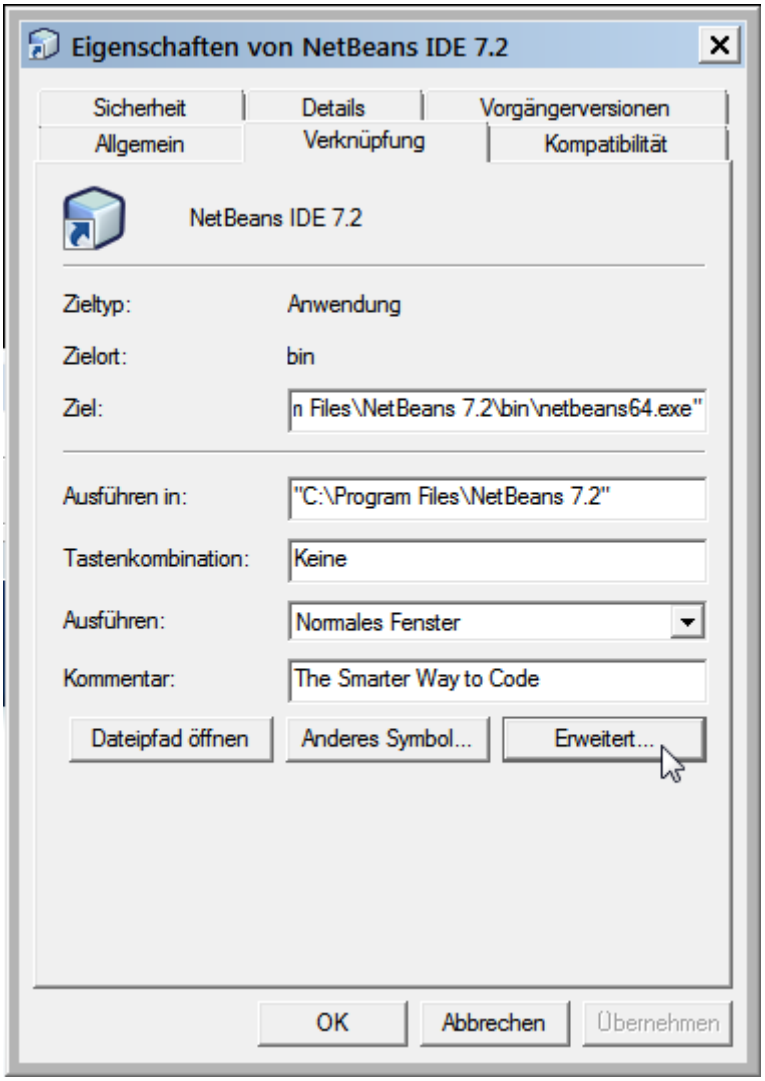
Grünes Dreieck für laufenden Server:



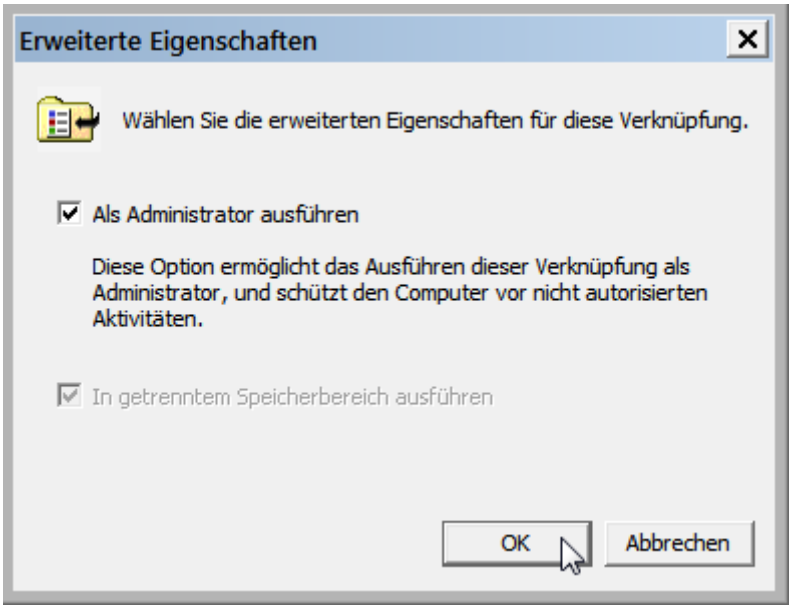
Falls es Probleme beim Deployen gibt, genauer eine Endlosaktion gestartet wird, kann dies an den Sicherheitseinstellungen des Rechners liegen. Eine häufiger funktionierende Lösung ist es, NetBeans mit Administrator-Rechten laufen zu lassen. Dazu macht man ein Rechtsklick auf dem Icon und wählt „Eigenschaften“.



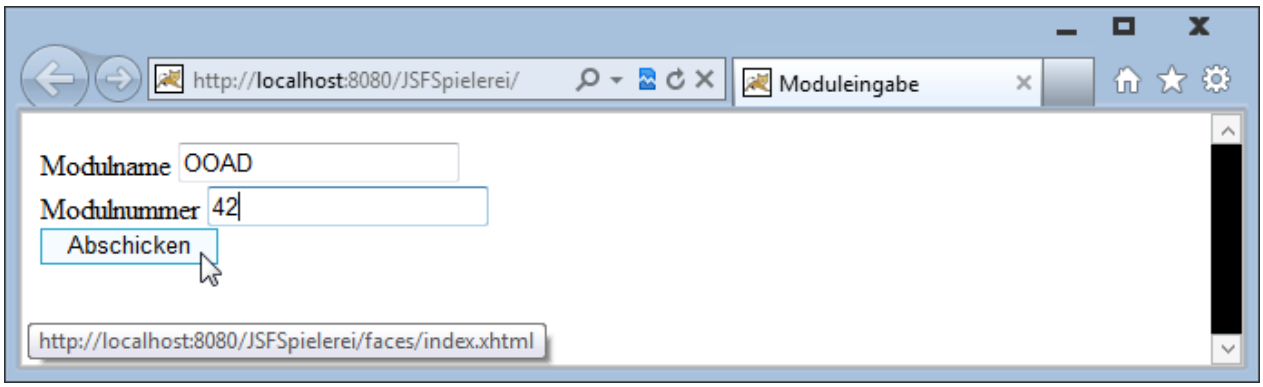
Man klickt auf „Erweitert…“.



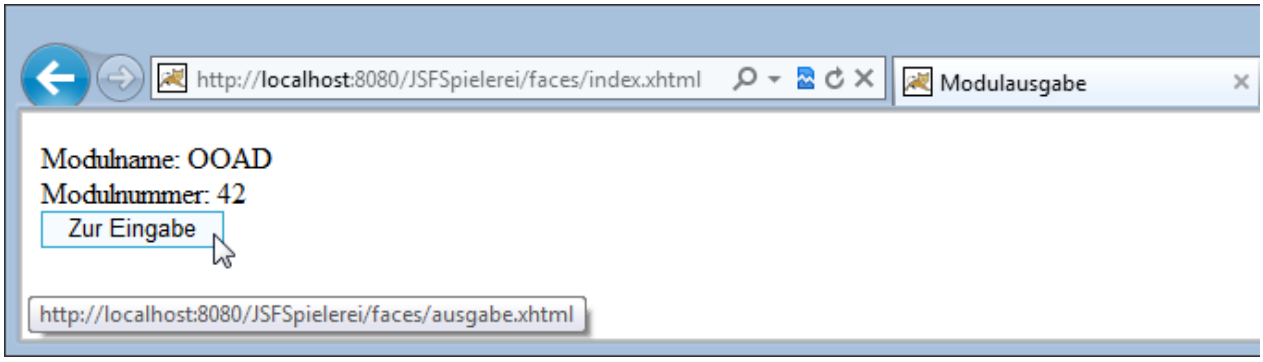
Man setzt den Haken bei „Als Administrator ausführen“ und bestätigt die Änderungen und etwaige Nachfragen mit „OK“.



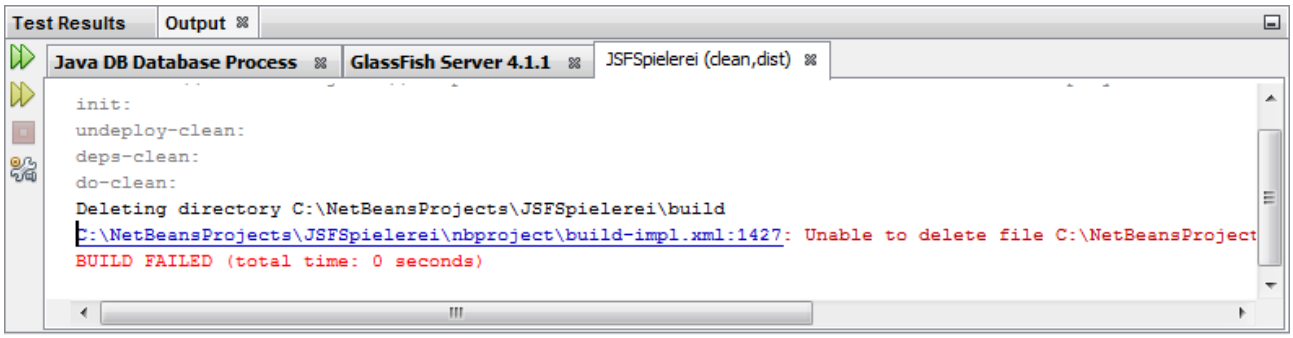
Nach dem Deployen und Starten der Applikation sieht man folgende Ausgabe im Default-Browser, der unter „Tools -> Options -> General“ eingestellt werden kann. Man sieht im folgenden Bild, dass der Server unter der Portnummer 8080 läuft, einige Werte eingetragen wurden und auf „Abschicken“ geklickt wird.



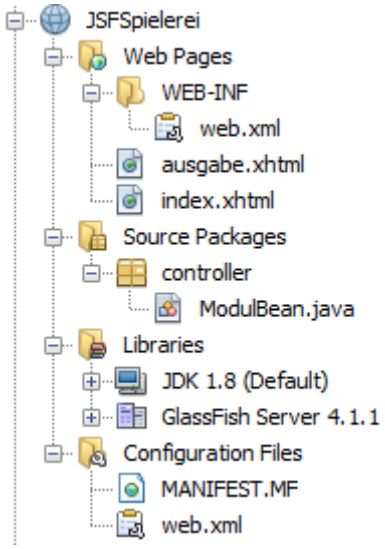
Erfüllen die Eingaben die angegebenen Randbedingungen, wird zur Ausgabeseite gewechselt.



Sollte es bei einer neuen Übersetzung zu folgender Fehlermeldung kommen, sei auf „14.2 Meldung „BUILD FAILED““ hingewiesen.

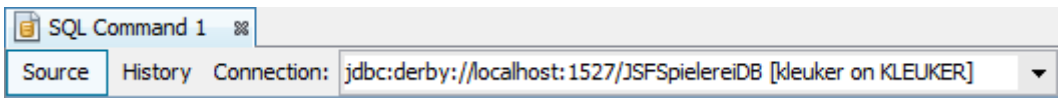


Das endgültige Projekt sieht wie folgt aus.



10.4 Einbindung von JPA mit EJB-Hilfe

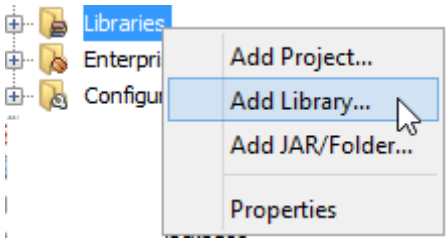
Es wird davon ausgegangen, dass eine Datenbank JSFSpielereiDB, wie in „6.2 Anlegen einer neuen Datenbank“ beschrieben, angelegt ist, die noch keinen Inhalt hat. Weiterhin kann der Apache Tomcat ohne Erweiterung, z. B. Apache TomEE, nicht genutzt werden, so dass hier der Fokus auf GlassFish liegt.



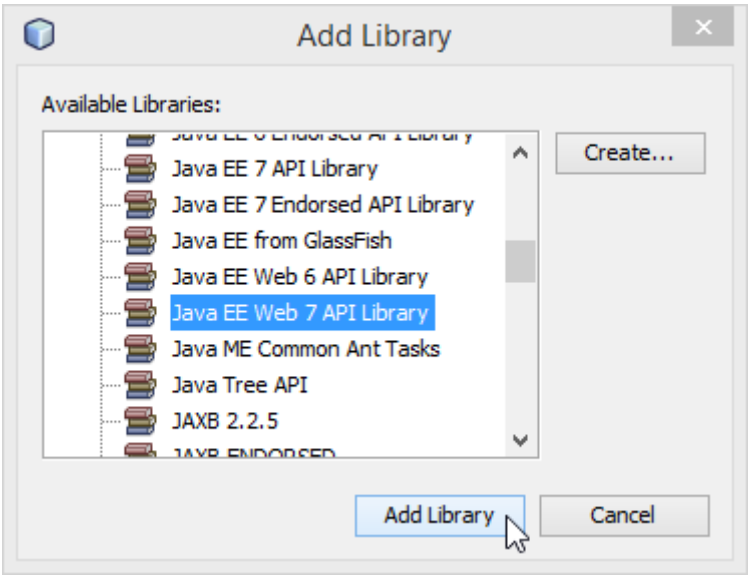
Zu Beginn existiert ein JSF-Projekt JSFSpielerei.

Nur bis NetBeans 8.0.2:

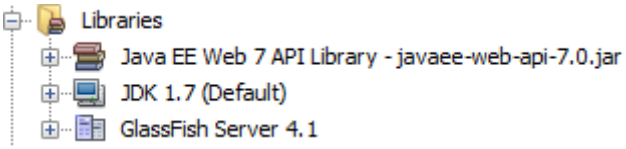
Bei dem Projekt wird die „Java EE Web 7 Library“ hinzugefügt. Nach einem Rechtsklick auf „Libraries“ wird „Add Library…“ geklickt.



Es wird die Bibliothek „Java EE Web 7 API Library“ ausgewählt und über „Add Library“ hinzugefügt.

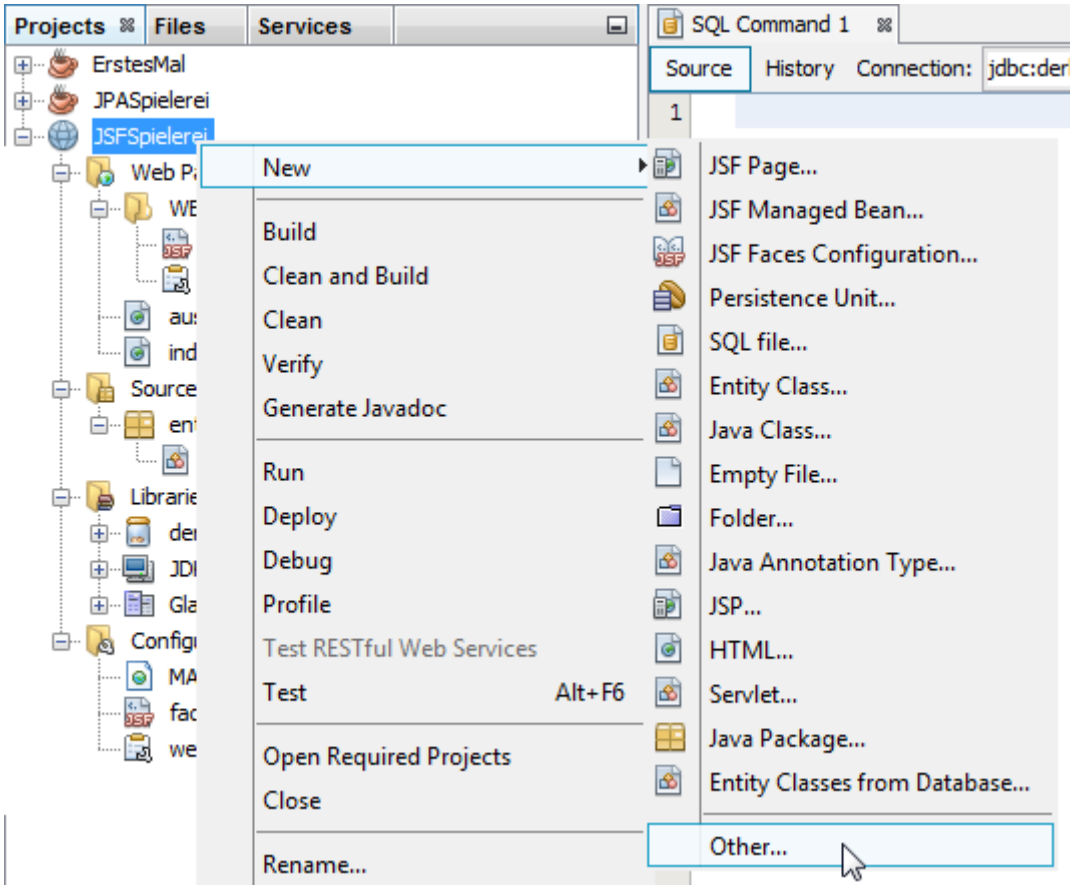


Das Library-Verzeichnis sieht dann wie folgt aus.

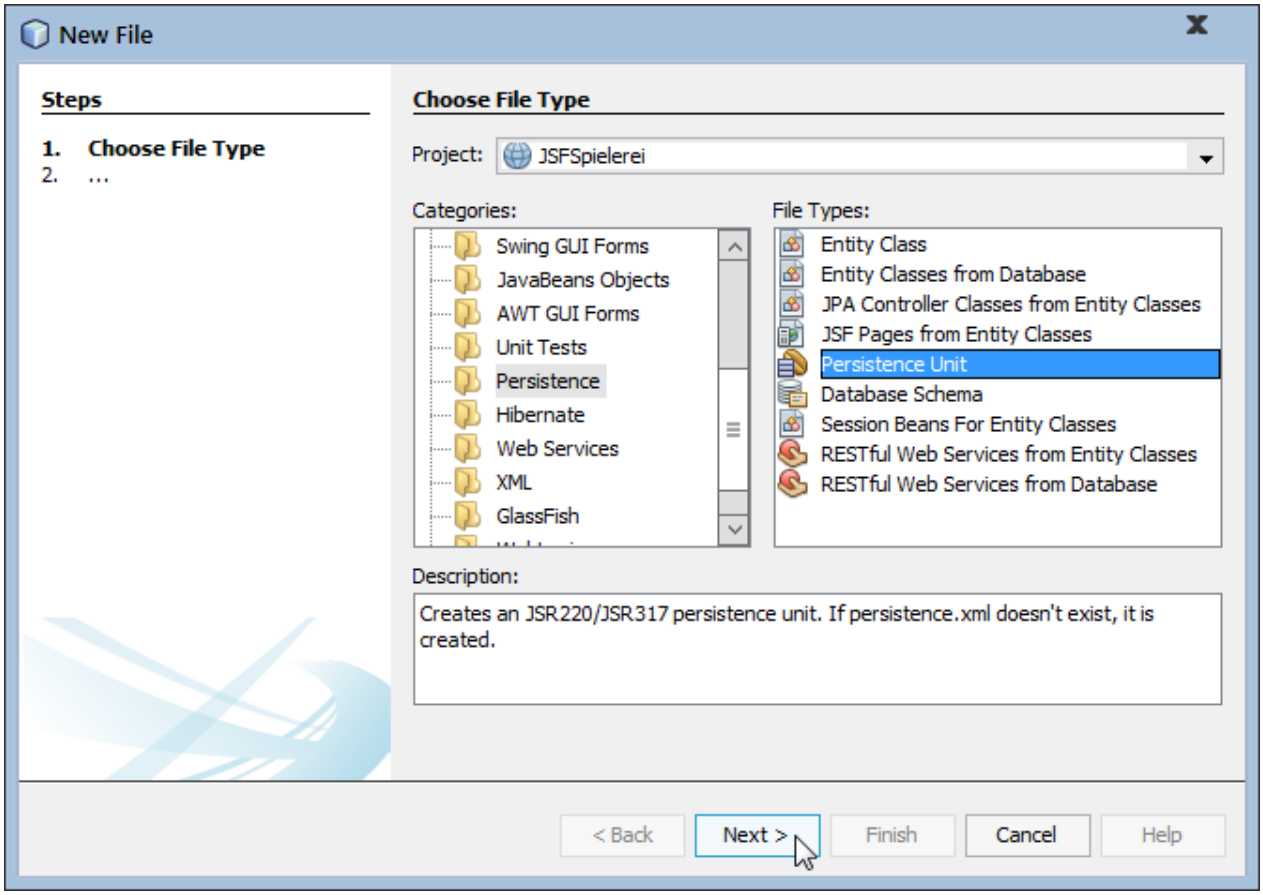


10.4.1 Einrichten der Persistence Unit

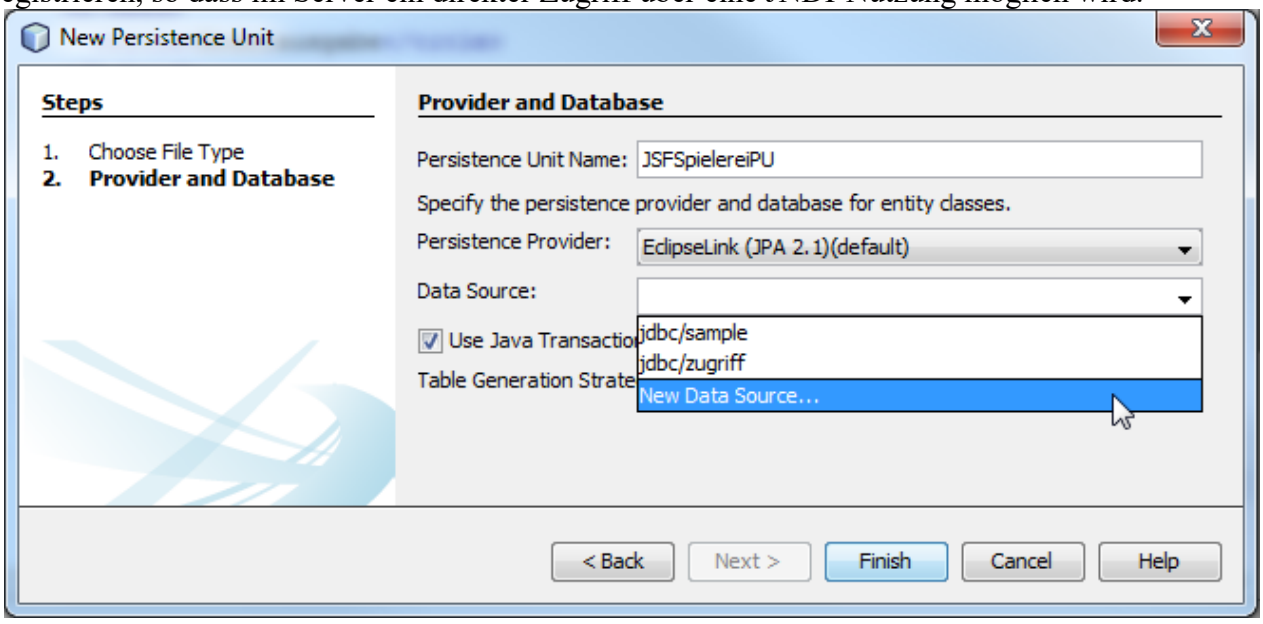
Um die DB-Nutzung zu ergänzen, wird zunächst ein Rechtsklick auf dem Projekt gemacht und „New“ sowie „Other...“ gewählt.



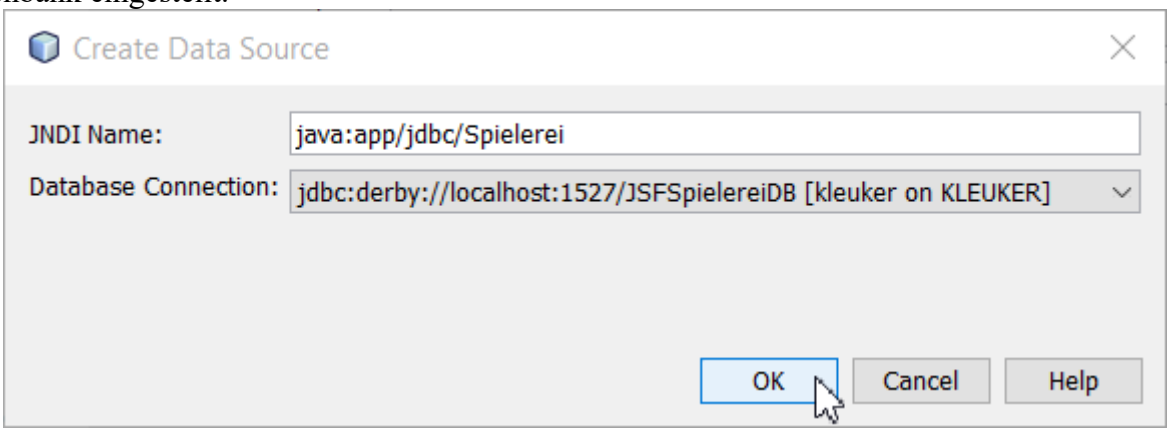
Danach wird links „Persistence“ und rechts „Persistence Unit“ ausgewählt.



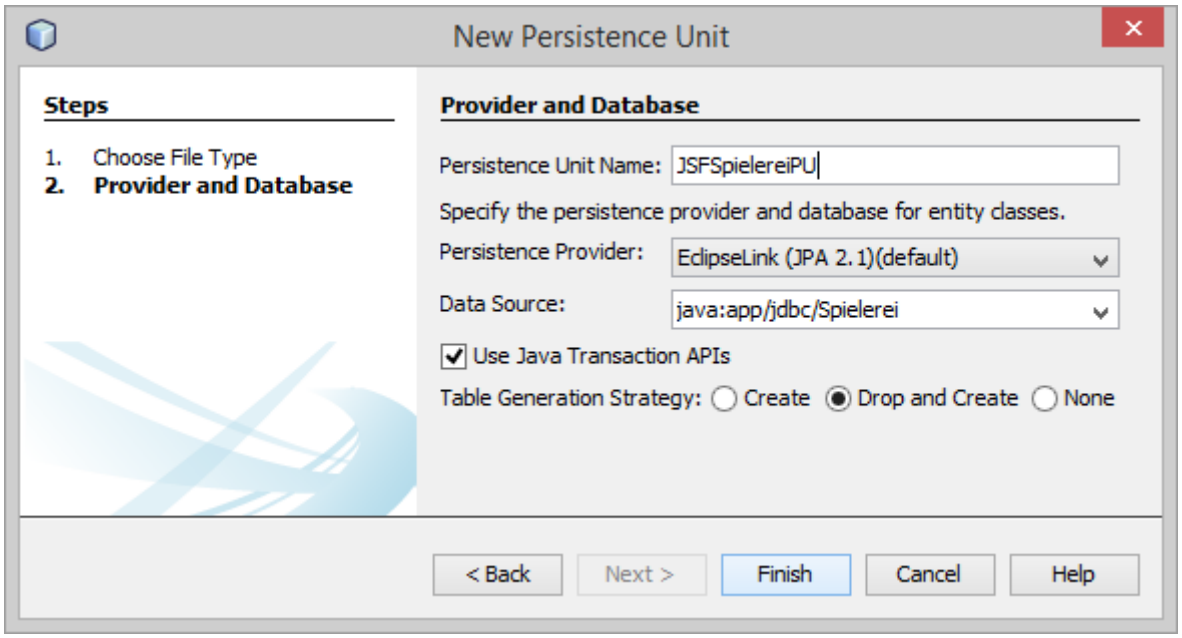
Der Name der Persistence Unit und der „Persistence Provider“ werden beibehalten. Jetzt wird der Pfeil rechts von „Data Source“ angeklickt und „Create New Datasource“ ausgewählt. Mit dem folgenden Schritt können wir unsere vorher angelegte Datenbank als Data Source des Servers registrieren, so dass im Server ein direkter Zugriff über eine JNDI-Nutzung möglich wird.



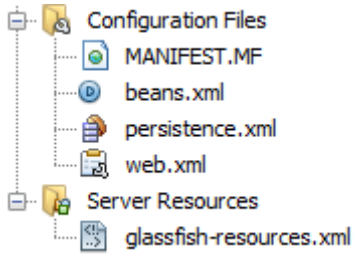
Der JNDI-Name muss mit „java:app/“ beginnen, wir dies ab NetBeans 8.1 vergessen, wird der Prefix „java:app/“ automatisch ergänzt. Bei der „Database Connection“ wird die vorher angelegte Datenbank eingestellt.



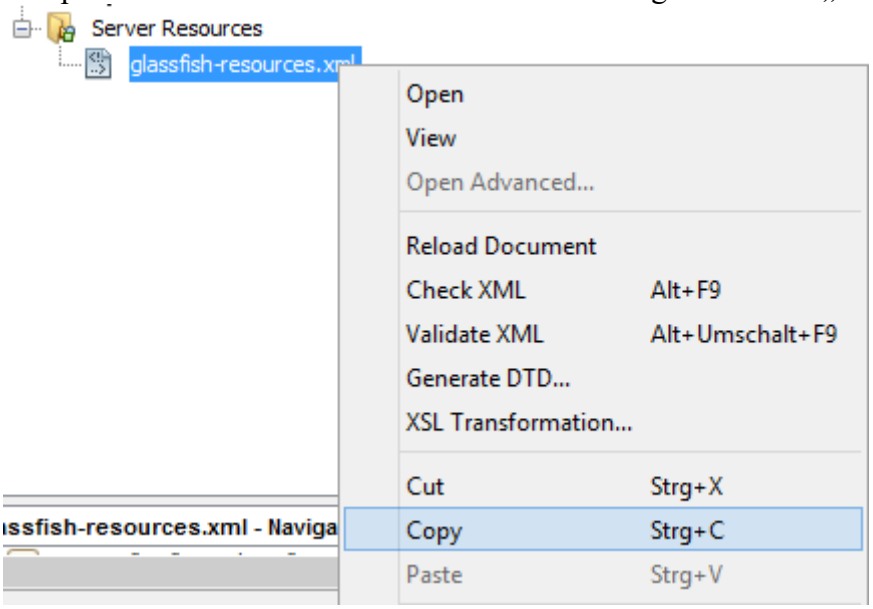
Die „Table Generation Strategy“ wird auf den gewünschten Wert, für die Entwicklung meist „Drop and Create“, gesetzt und danach das Anlegen der Persistence Unit mit Finish abgeschlossen. Der frei wählbare Namen der Persistence Unit wird später benötigt.



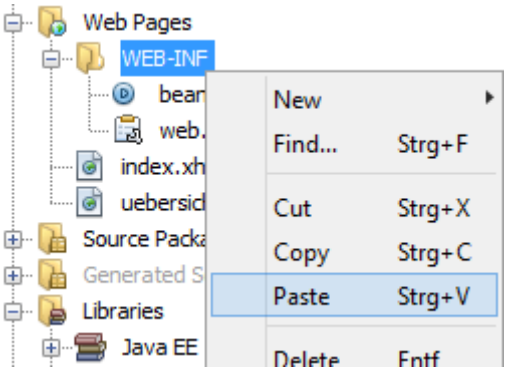
Ergänzung für NetBeans 8.0.1/8.0.2, ab 8.1 nicht benötigt: Die gerade vorgenommen Einstellungen werden in NetBeans aktuell in einem neu entstehenden Ordner Server Resources abgespeichert. Die Einstellungen liegen in der Server-individuellen Datei glassfish-resources.xml.



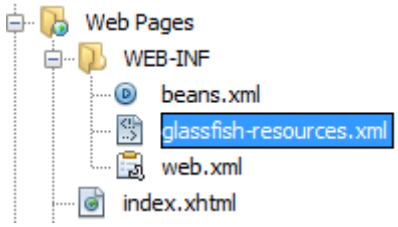
Diese Datei muss allerdings nach dem Standard im Ordner WEB-INF liegen, was eventuell in neueren Version bereits automatisch passieren sollte. Im konkreten Fall wird die Datei einfach in den passenden Ordner kopiert. Dazu wird ein Rechtsklick auf der Datei gemacht und „Copy“ ausgewählt.



Danach wird mit einem Rechtsklick auf dem Ordner WEB-INF die Datei mit „Paste“ eingefügt.



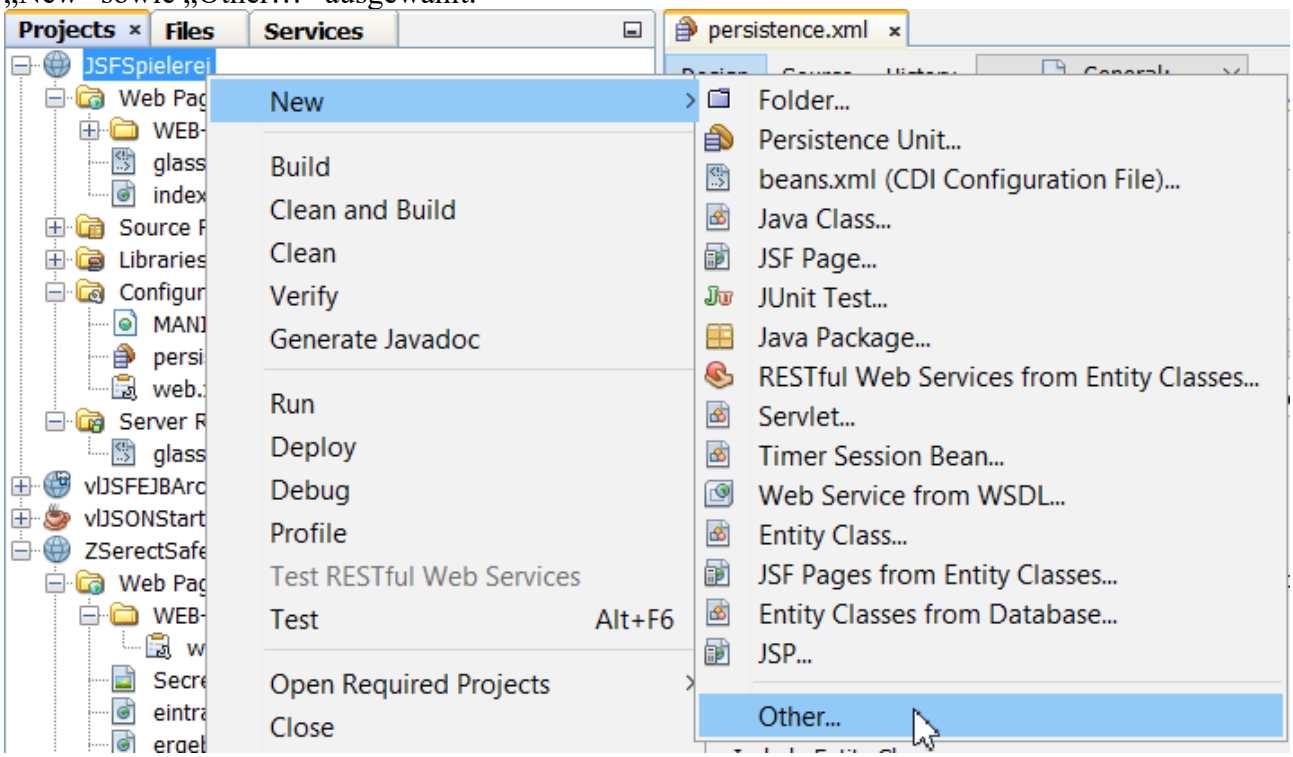
Das Endergebnis sieht wie folgt aus.



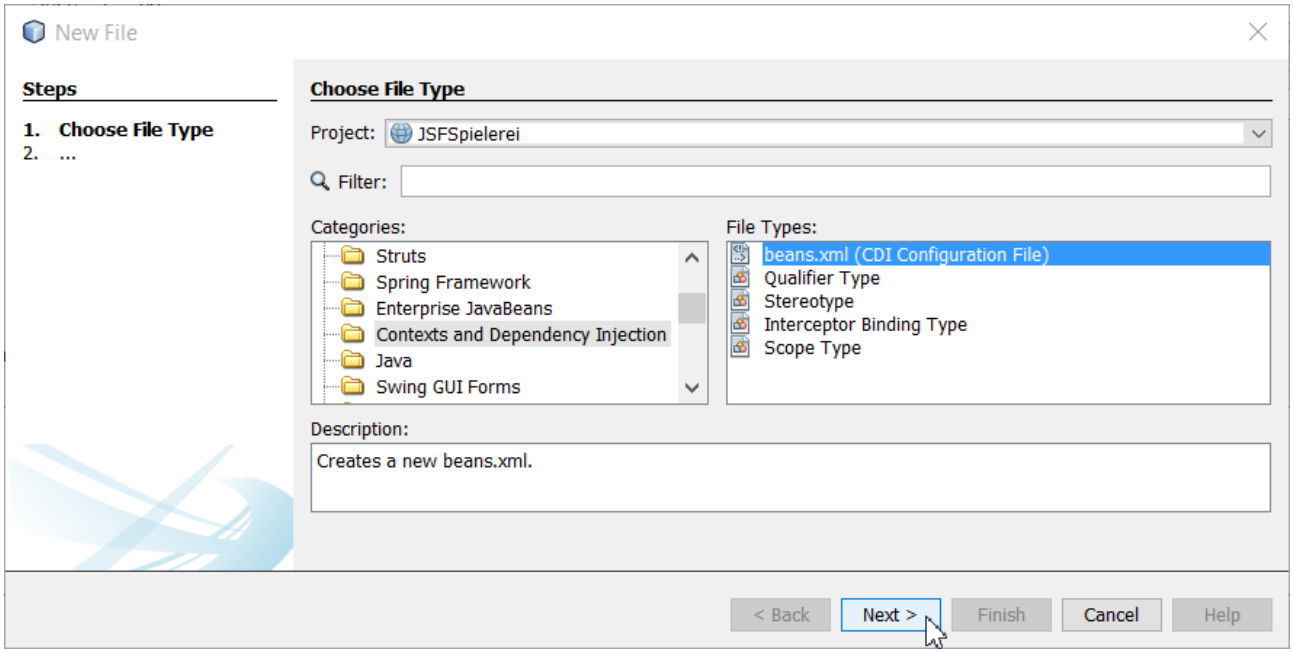
Man beachte, dass Änderungen in der persistence.xml, die die Data Source betreffen, immer in der Datei glassfish-resources.xml im falschen Ordner gespeichert werden, so dass die Datei nach der Änderung erneut in den richtigen Ordner zu kopieren ist. Dazu ist die alte Datei vorher zu löschen.

10.4.2 Anlegen der beans.xml

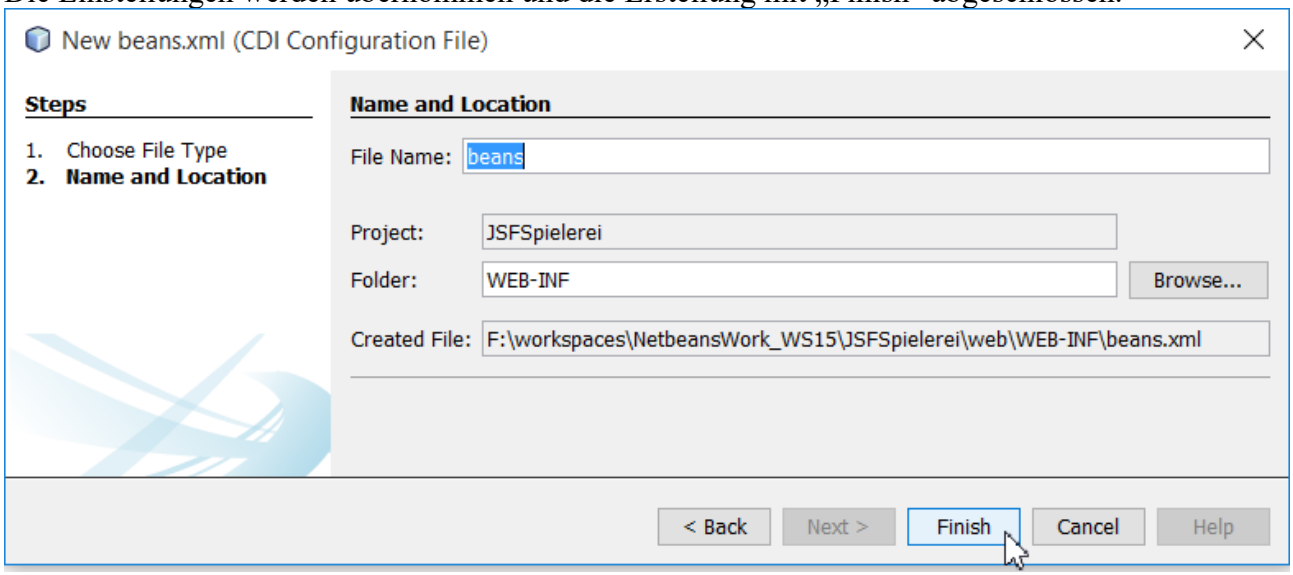
Zur Nutzung von CDI (Contexts and Dependency Injection) wird eine Datei beans.xml benötigt, die CDI-Einstellungen konfiguriert. Zur Erstellung wird ein Rechtsklick auf dem Projekt gemacht und „New“ sowie „Other…“ ausgewählt.



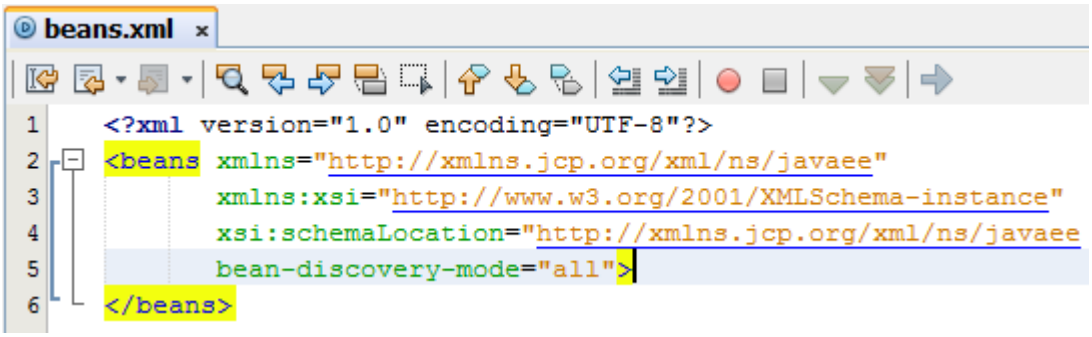
Links wird „Contexts and Dependency Injection“, rechts „beans.xml (CDI Configuration File) und dann „Next >“ geklickt.



Die Einstellungen werden übernommen und die Erstellung mit „Finish“ abgeschlossen.



Es öffnet sich die Datei beans.xml. In der fünften Zeile steht der Wert „annotated“, der hier auf „all“, wie in der nachfolgenden Abbildung gezeigt, geändert wird.



10.4.3 Erstellung der Entity-Klasse

Damit die Daten mit Hilfe von JPA persistiert werden können, wird eine Klasse Modul angelegt, die zur Persistierung genutzt wird. Details dazu können „7.2 Neue Entitys anlegen“ entnommen werden, das Ergebnis sieht wie folgt aus. Natürlich kann die Klasse auch direkt programmiert werden.

```
package entities;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Version;

@Entity
public class Modul implements Serializable {

  private static final long serialVersionUID = 2L;
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private int id;
  //@Column(unique=true)
  private int nr; // so sind gleiche Nummern erlaubt
  private String name;
  @Version int version;

  public Modul() {
  }

  public Modul(int nr, String name) {
    this.nr = nr;
    this.name = name;
  }

  ///*
  public int getVersion() {
```

```
    return version;
  }

  public void setVersion(int version) {
    this.version = version;
  }
//*/

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public int getNr() {
    return nr;
  }

  public void setNr(int nr) {
    this.nr = nr;
  }

  @Override
  public int hashCode() {
    int hash = 0;
    hash += (int) id;
    return hash;
  }

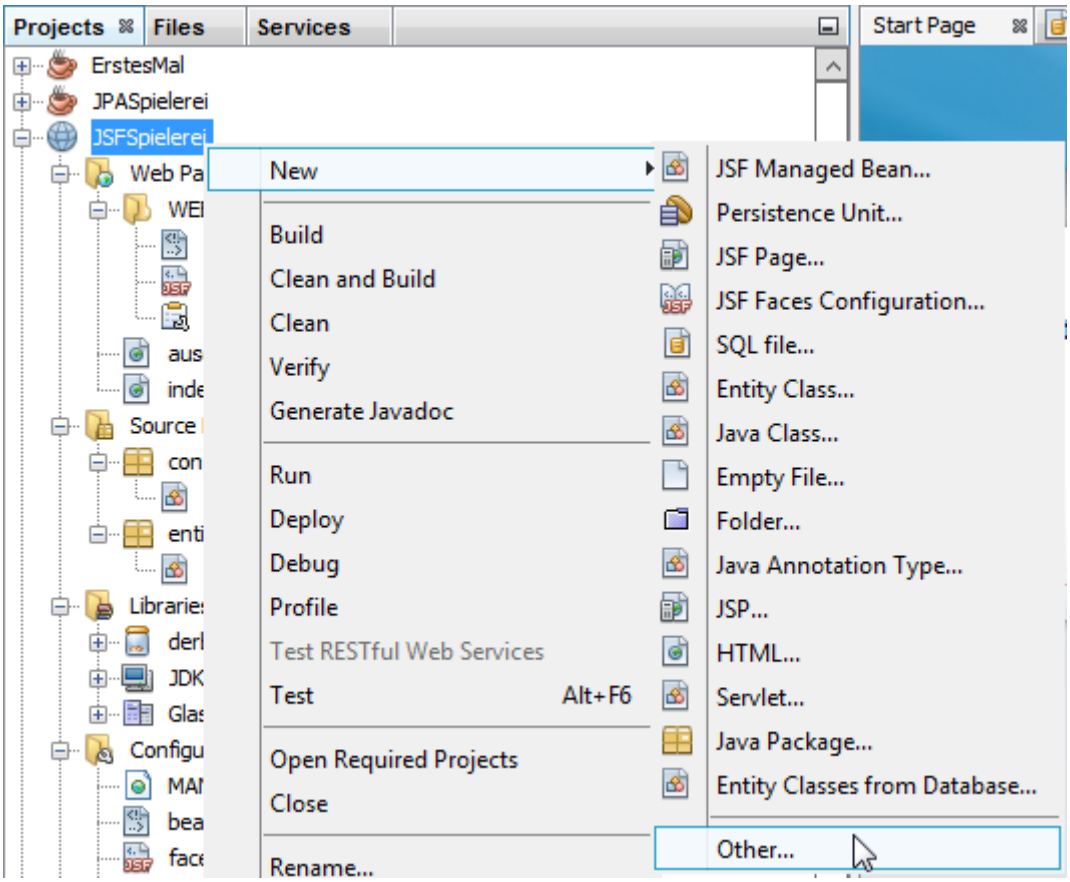
  @Override
  public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    if (!(object instanceof Modul)) {
      return false;
    }
    Modul other = (Modul) object;
    if (this.id != other.id) {
      return false;
    }
```

```
        return true;
    }

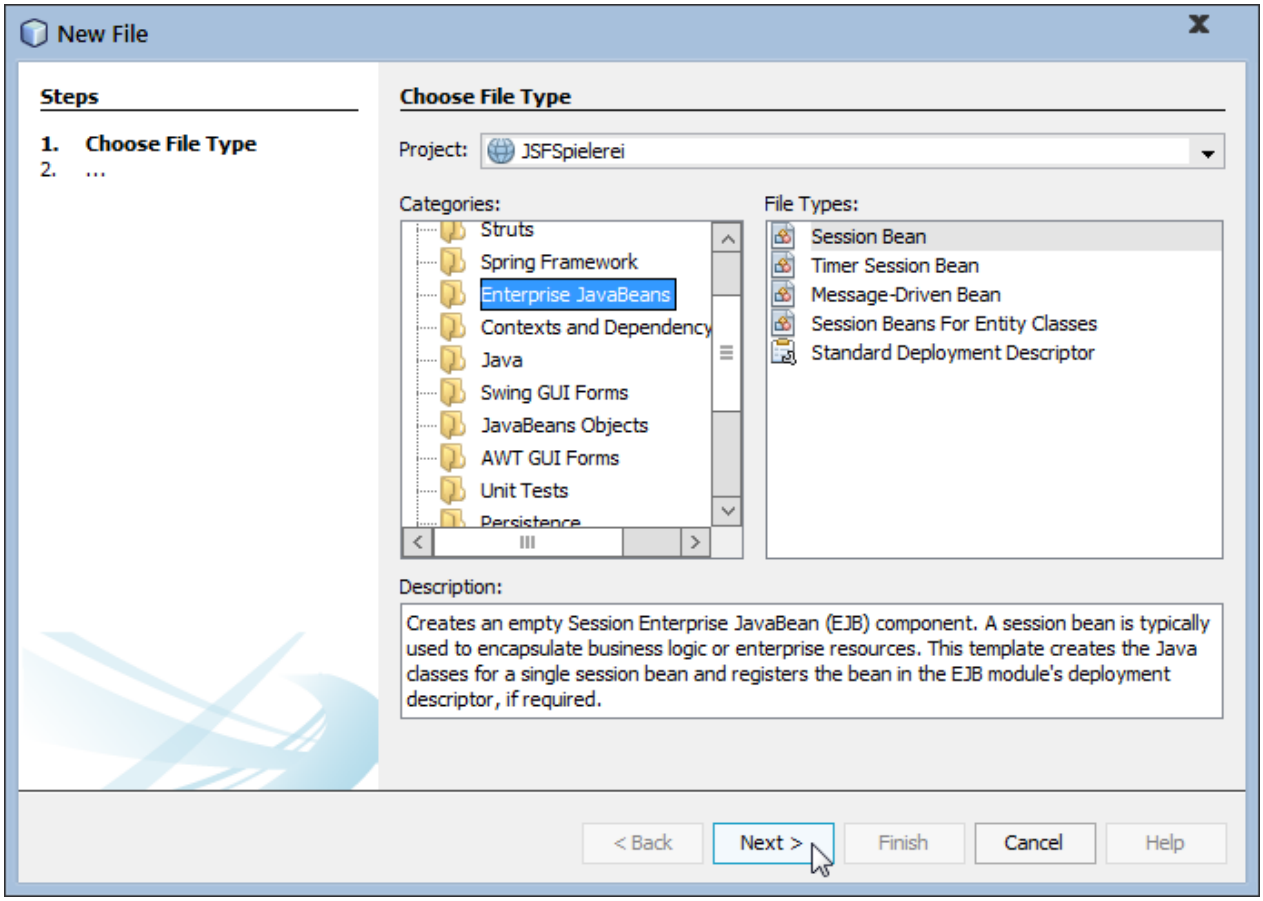
    @Override
    public String toString() {
        return "entities.Moudul[id=" + id + "]";
    }
}
```

10.4.4 Anlegen der EJB

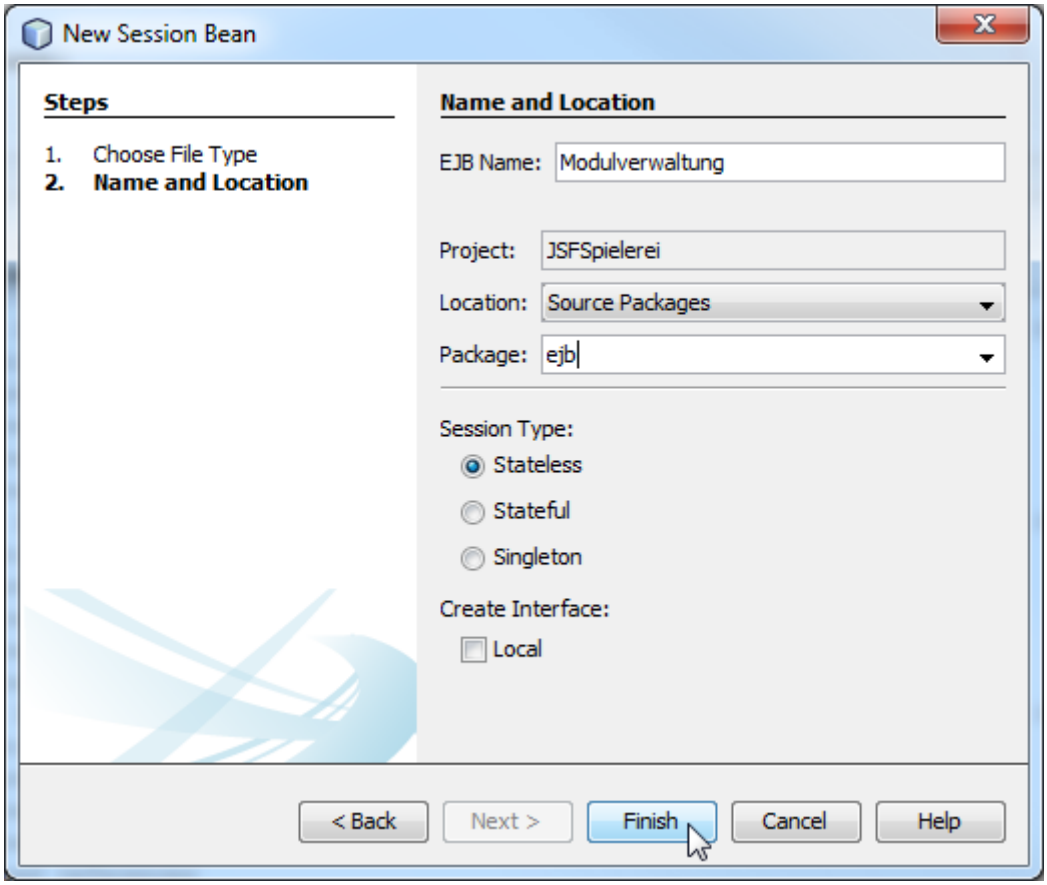
Die eigentliche Datenbankanbindung soll durch eine Enterprise Java Bean (EJB), genauer eine lokale Stateless Session Bean, erfolgen. Der wesentliche Vorteil der EJBs ist, dass sie automatisch in einer eigenen Transaktion ablaufen und man sich so nicht um die Transaktionssicherheit kümmern muss. Die Bean kann direkt programmiert werden, das Anlegen wird aber auch die NetBeans unterstützt. Zum Anlegen der EJB wird ein Rechtsklick auf das Projekt gemacht und „New -> Other…“ ausgewählt.



Links wird „Enterprise JavaBeans“ angeklickt und rechts „Session Bean“ ausgewählt.



Es wird ein Name und ein Paket eingetragen, Stateless gewählt und auf ein Interface verzichtet, da diese EJB nur von unserer Applikation angesprochen wird.



Der weitere Programmcode wird manuell ergänzt und kann wie folgt aussehen. Die hier genutzte JPQL-Anfrage sollte besser als Named Query in der Entitätsklasse stehen.

```
package ejb;

import entities.Modul;
import java.util.List;
import javax.ejb.Stateless;
import javax.inject.Inject;
import javax.persistence.EntityManager;

@Stateless
public class Modulverwaltung {
    @Inject //@RealClass
    private EntityManager em;

    public void persist(Object object) {
        em.persist(object);
    }

    public List<Modul> getModule() {
        return em.createQuery("SELECT m FROM Modul m", Modul.class)
                .getResultList();
```

```
        }
    }
```

Ohne weiter auf Details einzugehen wird mit der @Inject-Annotation die Umgebung, hier der Server, genauer der EJB-Container, angewiesen, ein passendes Objekt zur Verfügung zu stellen. Da es für das angegebene Interface keine standardisierte Erstellung gibt, muss ein Erzeuger spezifiziert werden. Dies erfolgt im Beispiel mit der Klasse MyProducers, in der die Erzeugung mit @Produces annotiert wird. Die Klasse sieht wie folgt aus, der Name aus der Persistence Unit wurde übertragen.

```
package ejb;

import javax.enterprise.inject.Produces;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

public class MyProducers {
    @Produces //@RealClass
    @PersistenceContext(unitName = "JSFSpielereiPU")
    private EntityManager em;
}
```

Der hier skizzierte Ansatz ermöglkich es, die EJB unter die Kontrolle von CDI (Contexts and Dependency Injection) zu stellen. Hier sind andere Ansätze möglich.

10.4.5 Einbindung der EJB in der ManagedBean mit Injektion

Diese EJB muss jetzt in der Verwaltungsklasse für die Web-Seiten genutzt werden. Hierbei wird wieder Dependency Injection genutzt. Die Klasse sieht dann wie folgt aus, die Ausnahmebehandlung sollte noch ausgebaut werden. Mit @Inject wird die Stateless Session Bean eingebunden. Mit @PostConstruct wird eine Methode annotiert, die garantiert nach Erstellung und vor der weiteren Benutzung aufgerufen werden soll. Die Methode „ersetzt“ damit den Konstruktor. Diskutabel für größere Ansätze ist die hier genutzt Idee direkt ein Entity-Objekt zur Bearbeitung zu nutzen, unabhängiger wäre die Nutzung von Exemplarvariablen pro Eingabefeld. Weiterhin sollten alle Konstanten in deine getrennte Klasse ausgelagert werden.

```
package controller;

import ejb.Modulverwaltung;
import entities.Modul;
import java.io.Serializable;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.enterprise.context.RequestScoped;
import javax.inject.Inject;
import javax.inject.Named;

@Named("module")
@RequestScoped
public class ModulBean implements Serializable {
```

```
private static final long serialVersionUID = 1L;
@Inject
private Modul modul;
@Inject
private Modulverwaltung mv;

public ModulBean() {
}

@PostConstruct // Spielerei
public void modulGarantieren() {
    if (this.mv.getModule().isEmpty()) {
        Modul m = new Modul(42, "Bachelorarbeit");
        this.mv.persist(m);
    }
}

// da get-Methode, ist Zugriff ueber module,module moeglich
public List<Modul> getModule() {
    return this.mv.getModule();
}

public String uebernehmen() {
    try {
        mv.persist(modul);
    } catch (Exception e) {
        // zu kurz
    }
    return "ausgabe.xhtml";
}

public String eingeben() {
    return "index.xhtml";
}

public String anzeigen() {
    return "ausgabe.xhtml";
}

public Modul getModul() {
    return modul;
}

public void setModul(Modul modul) {
    this.modul = modul;
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 67 * hash + (this.modul != null ? this.modul.hashCode() : 0);
```

```
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final ModulBean other = (ModulBean) obj;
        if (this.modul != other.modul && (this.modul == null
                || !this.modul.equals(other.modul))) {
            return false;
        }
        return true;
    }
}
```

Die Eingabeseite index.xhtml sieht wie folgt aus.

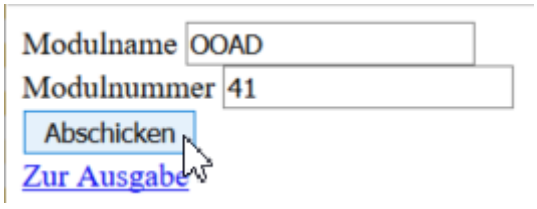
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Moduleingabe</title>
  </h:head>
  <h:body>
    <h:form id="index">
      <h:outputText value="Modulname "/>
      <h:inputText id="mname" value="#{module.modul.name}"
                   required="true" /><br/>
      <h:outputText value="Modulnummer "/>
      <h:inputText id="mnr" value="#{module.modul.nr}"
                   required="true"/><br/>
      <h:commandButton id="abschicken" value="Abschicken"
                       action="#{module.uebernehmen}"/><br/>
      <h:commandLink id="ausgabelink" value ="Zur Ausgabe"
                     action="#{module.anzeigen}" immediate="true"/>
    </h:form>
  </h:body>
</html>
```

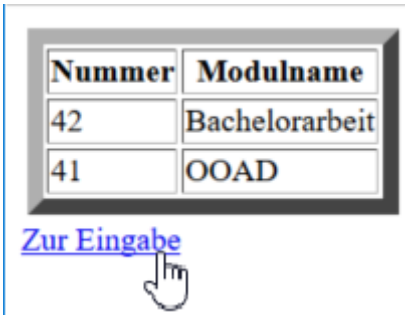
Als letztes muss die Ausgabe noch um die Möglichkeit ergänzt werden, die Liste aller Module anzuzeigen. Die Datei ausgabe.xhtml sieht wie folgt aus.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

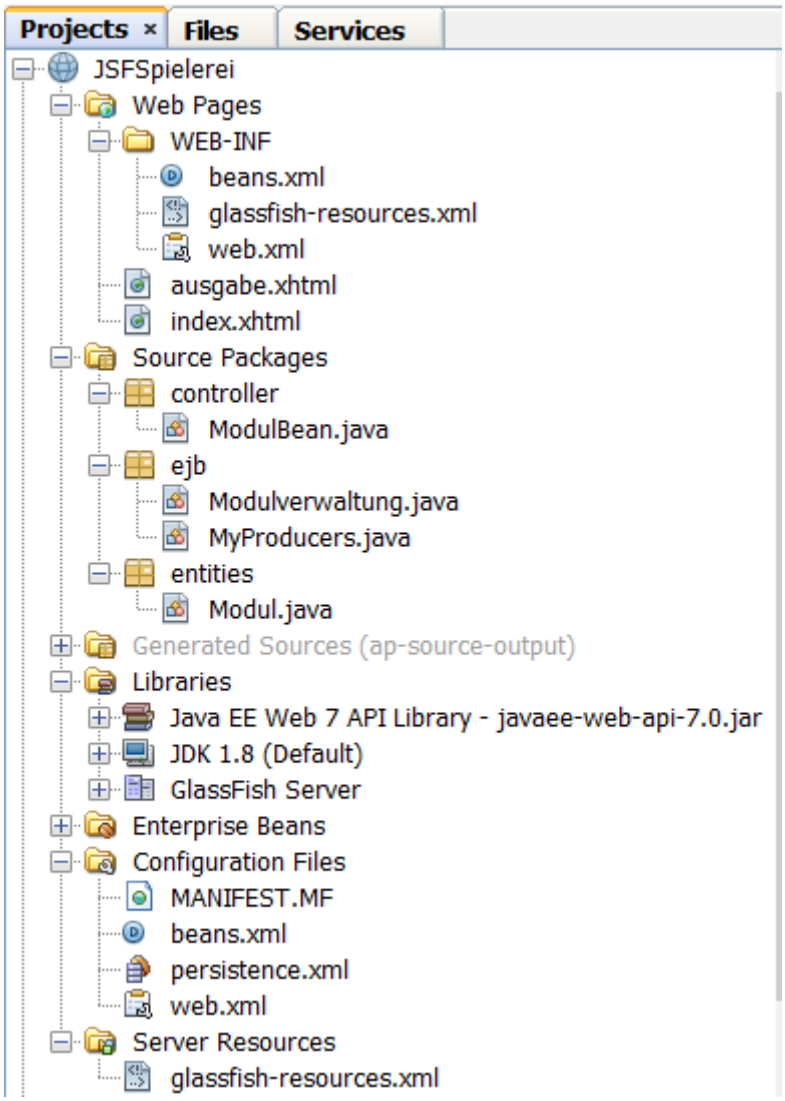
```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <title>Modulausgabe</title>
    </h:head>
    <h:body>
        <h:form id="ausgabe">
            <h:messages globalOnly="true"/>
            <h:panelGrid rendered="#{!empty module.module}">
                <h:dataTable id="tabelle" value="#{module.module}" var="m" border="8"
                             frame="box" >
                    <h:column >
                        <f:facet name="header">
                            <h:outputText value="Nummer" />
                        </f:facet>
                        <h:outputLabel value="#{m.nr}"/>
                    </h:column>
                    <h:column>
                        <f:facet name="header">
                            <h:outputText value="Modulname" />
                        </f:facet>
                        <h:outputLabel value="#{m.name}"/>
                    </h:column>
                </h:dataTable>
            </h:panelGrid>
            <h:commandLink id ="zurueck" value="Zur Eingabe" action="index"/>
        </h:form>
    </h:body>
</html>
```

Nach dem Programmstart kann die Applikation genutzt werden. Die folgenden Abbildungen zeigen einen Beispielablauf.





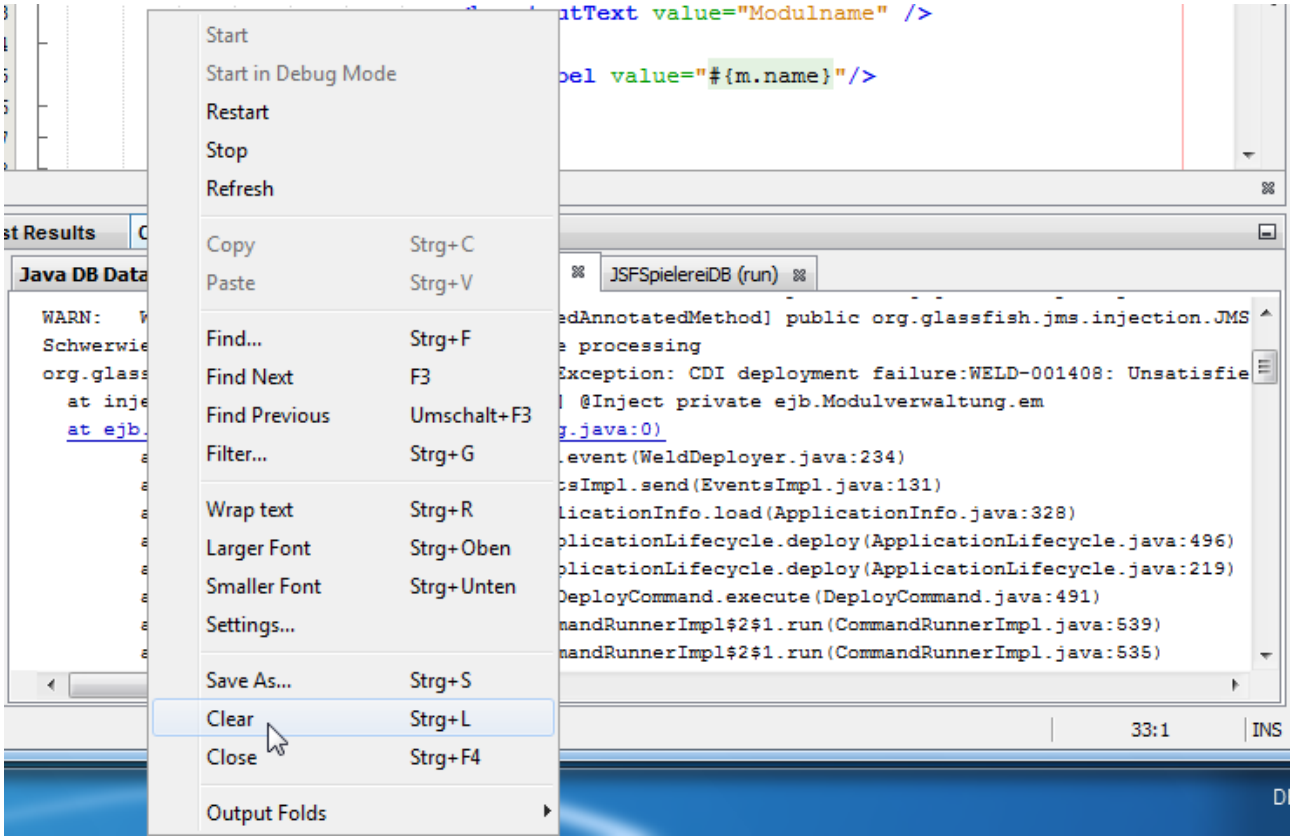
Das resultierende Projekt sieht wie folgt aus.



Bei auftretenden Fehlern ist das nachfolgende Unterkapitel abzuarbeiten, zunächst aber auf eine fehlende „beans.xml“ und den dortigen Eintrag „all“ zu prüfen.



Detaillierte Fehlermeldungen stehen unter dem Reiter „GlassFish Server 4.1.1“. Da diese Informationen immer fortgeschrieben werden, ist es vor einem erneuten Versuch sinnvoll, die Meldungen zu löschen. Dazu wird ein Rechtsklick gemacht und „Clear“ ausgewählt.

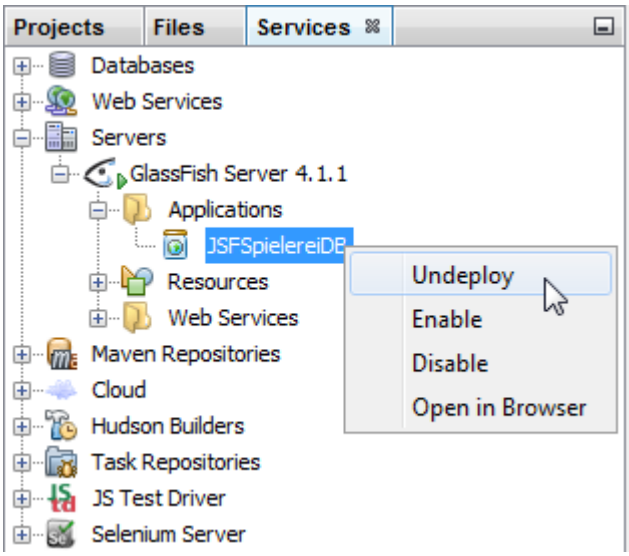


10.5 Check-Liste bei Fehlern

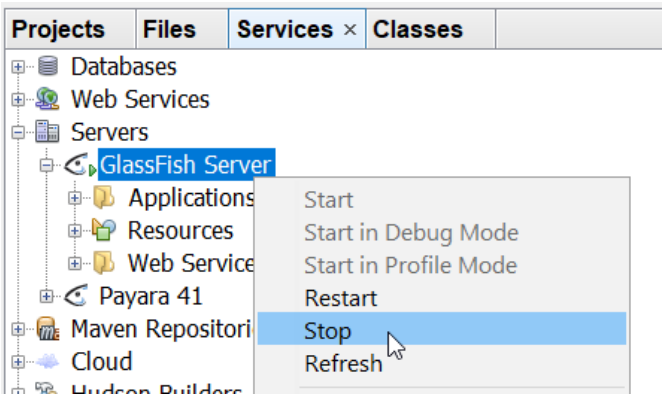
Web-Projekte mit einer Datenbankanbindung bieten bei der Erstellung eine große Möglichkeit Flüchtigkeitsfehler zu machen, die oft nur mit einer Meldung der Art „Computer sagt nein“ zum Start der Applikation führen. Die folgende, sicherlich nicht vollständige Liste sollte zunächst abgearbeitet werden.

1. Aufruf von „Clean & Build“, da bei wesentlichen Änderungen NetBeans nicht alle Schritte automatisch durchführt, danach Applikation wieder starten. Bei Löschproblemen das Unterverzeichnis „build“, wie in „14.2 Meldung „BUILD FAILED““ beschrieben, entfernen und dann wieder „Clean & Build“ durchführen. Generell den genutzten Browser vor dem Neustart schließen.
2. Generell sollte der benutzte Browser vor einem Neustart der Applikation immer geschlossen werden, damit keine Nutzung von alten Cache-Daten erfolgt.
3. Wurde eine Datei beans.xml zur CDI-Nutzung angelegt?
4. Wurde in der beans.xml bean-discovery-mode="all" gesetzt?
5. Wurde die aktuellste Version der Datei glassfish-resources.xml in den Ordner WEB-INF (nicht direkt Web Pages) kopiert, bzw. ab NetBeans 8.1 dort schon abgelegt?
6. Wurde die Datenbank ohne Nutzer oder Passwort angelegt? Dann kann es teilweise zu Zugriffsproblemen kommen; dann muss die Datenbank komplett gelöscht und wieder neu angelegt werden.
7. Wurden Entitätsklassen so verändert, dass eine Änderung der Tabellen notwendig ist? Ist das der Fall müssen die Tabellen zunächst von Hand mir Rechtsklick „Delete“ in der richtigen Reihenfolge gelöscht werden.
8. Sind die Entitätsklassen POJOs, haben sie einen parameterlosen Konstruktor, haben alle Objektvariablen getter und setter? Implementieren die Klassen Serializable? Haben die Klassen sinnvolle equals- und hashCode-Methoden, die typischerweise nur die id, also den Primary Key nutzen?
9. Wurden die richtigen Annotationen eingebunden, z. B. für @SessionScoped import javax.enterprise.context.SessionScoped;?
10. Steht bei der Annotation @PersistenceContext(unitName = "JSFSpielereiPU") der korrekte unitName aus der persistence.xml, wobei auch Groß- und Kleinschreibung beachtet wurde?
11. Wurde überhaupt eine Datei persistence.xml angelegt?
12. Wurde in der Stateless Session Bean korrekt @Inject private EntityManager em; genutzt? (Sollten verschiedene EntityManager-Objekte genutzt werden, sind diese eindeutig durch einen zusätzlichen CDI-Qualifier unterscheidbar?)
13. Ist in der persistence.xml nur eine Persistence Unit eingetragen? (Mehrere sind theoretisch möglich; es muss aber getrickst werden)
14. Wurde das Projekt mit einer älteren NetBeans- oder Java- oder JSF-Version erstellt, z. B. eine alte Lösung kopiert? (dann stimmen Einstellungen nicht; bester Weg: Projekt neu anlegen und beans.xml sowie persistence.xml neu erstellen, Web-Seiten und Java-Code kopieren)

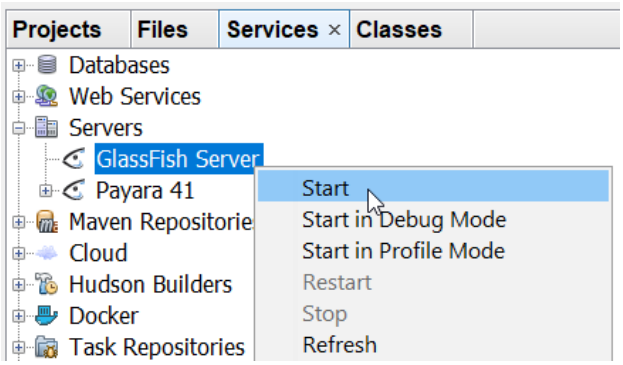
15. Wurde das Projekt mit einer neueren, bzw. anderen NetBeans-Version erstellt, dann kann es zu Problemen mit internen Einstellungen kommen. Die Lösung ein Projekt in der zu nutzenden NetBeans zu erstellen und den Quellcode (nur diesen) in das neue Projekt zu kopieren. Alle Konfigurationsdateien (beans.xml, persistence.xml, …) sind neu zu erstellen.
16. Probieren Sie aus, ob die Datenbank direkt über NetBeans geöffnet werden kann, ansonsten könnte ein Rechteproblem vorliegen (s. „6.9 AccessControlException beim Einspielen einer jar-Datei“)
17. In seltenen Fällen funktionieren Programme beim ersten Deploy-Versuch nicht, hier kann ein zweiter Start durchaus helfen.
18. Gerade bei kleinen Änderungen, wie einem neuen @Inject, scheint der Undeploy-Deploy-Zyklus nicht immer zu funktionieren, hier sollten zunächst alle Applikationen auf dem Server von Hand entfernt und dann der Server (oder NetBeans) neu gestartet werden. Zum Entfernen der Apps wird der Server unter „Services“ aufgeklappt, dann „Applications“ aufgeklappt, mit gerückter Strg-Taste die zu entfernenden Applikationen ausgewählt, darauf ein Rechtsklick gemacht und „Undeploy“ ausgewählt.



Dann wird ein Rechtsklick auf dem Server gemacht und „Stop“ geklickt. Es sollte kein „Restart“ genutzt werden, da dann leider anscheinend nicht alle temporären Datei gelöscht werden.



Danach ist „Start“ anwählbar und der Server wird so wieder gestartet.



19. Ein Neustart von NetBeans kann sinnvoll sein.
20. Eine Neuinstallation, bzw. weitere Installation des Glassfish-Servers, kann dessen Probleme beheben.
21. Nie, nie sollten Projekte neuer NetBeans-Versionen in einem älteren NetBeans geöffnet werden, da die Konfigurationsmöglichkeiten nicht zusammenpassen müssen. In dem Fall ist eine Neuinstallation von NetBeans notwendig. Beim ersten Start gilt wie immer, dass keine alten Einstellungen importiert werden dürfen.

10.6 Testen von JEE-Programmen mit Arquillian

Durch die Nutzung von CDI können einfache JUnit-Tests nicht alle Bereiche abdecken. Generell gilt aber, was mit JUnit relativ einfach zu testen ist, muss mit JUnit getestet werden. Dies gilt auch für einfache @Inject, da die zugehörigen Objekte über eine set-Methode injiziert werden können. Weiterhin ist bei komplexeren injizierten Objekten über Mocks nachzudenken, wie sie z. B. durch Mockito (http://site.mockito.org/) leicht zu erstellen sind.

Für komplexere Tests steht Arquillian (http://arquillian.org/) zur Verfügung, dessen Konzepte hier nur kurz skizziert werden sollen. Arquillian nutzt zum Testen einen JEE-Server, der allerdings nicht dem genutzten Server entsprechen muss, was insbesondere die Installation erleichtert und zu wesentlichen besseren Startzeiten führt. Reale Server sind auch nutzbar, hier wird nur die embedded Version des Glassfish-Servers genutzt, der sich im Verhalten sehr ähnlich zur Standard-Variante verhält.

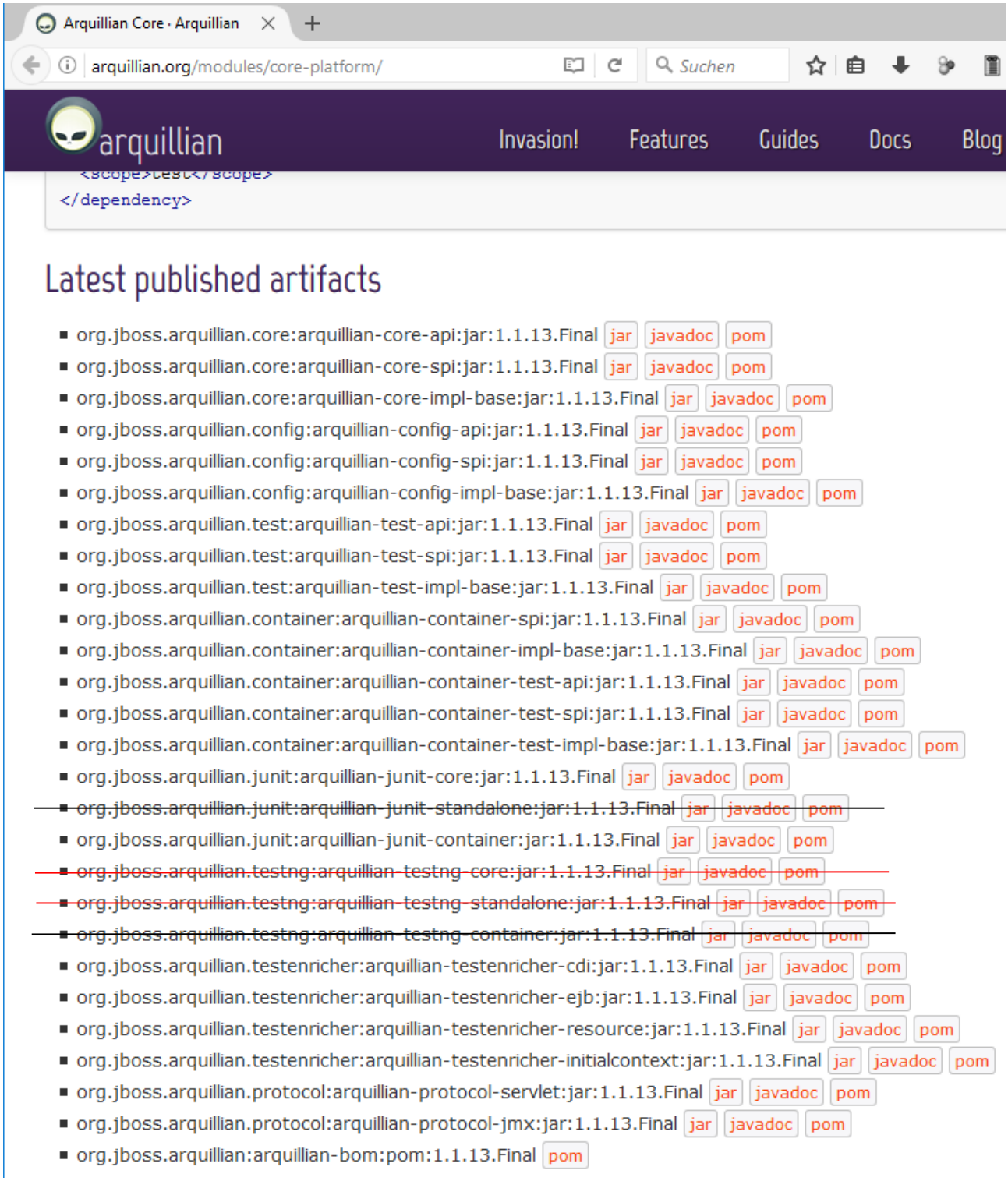
Arquillian wird nach Dokumentation generell mit einem Build-Werkzeug, üblicherweise Maven

(https://maven.apache.org/) installiert, was in Unternehmen ein Standardprozess ist, hier aber aus Komplexitätsgründen für Anfänger nicht betrachtet wird. Da die Tests einen eigenen Server nutzen, werden sie in einem eigenen Projekt, genauer in einem einfachen Java-Projekt, ausgeführt. Generell unterstützt Arquillian mit dem integrierten Paket ShrinkWrap die Konstruktion von Web-Applikationen aus einzelnen Class- und benötigten Konfigurationsdateien. Mit dem embeddedGlassfish-Server besteht aber die weit einfachere Möglichkeit, das von NetBeans erstellte Projekt als war-Datei direkt zu deployen.

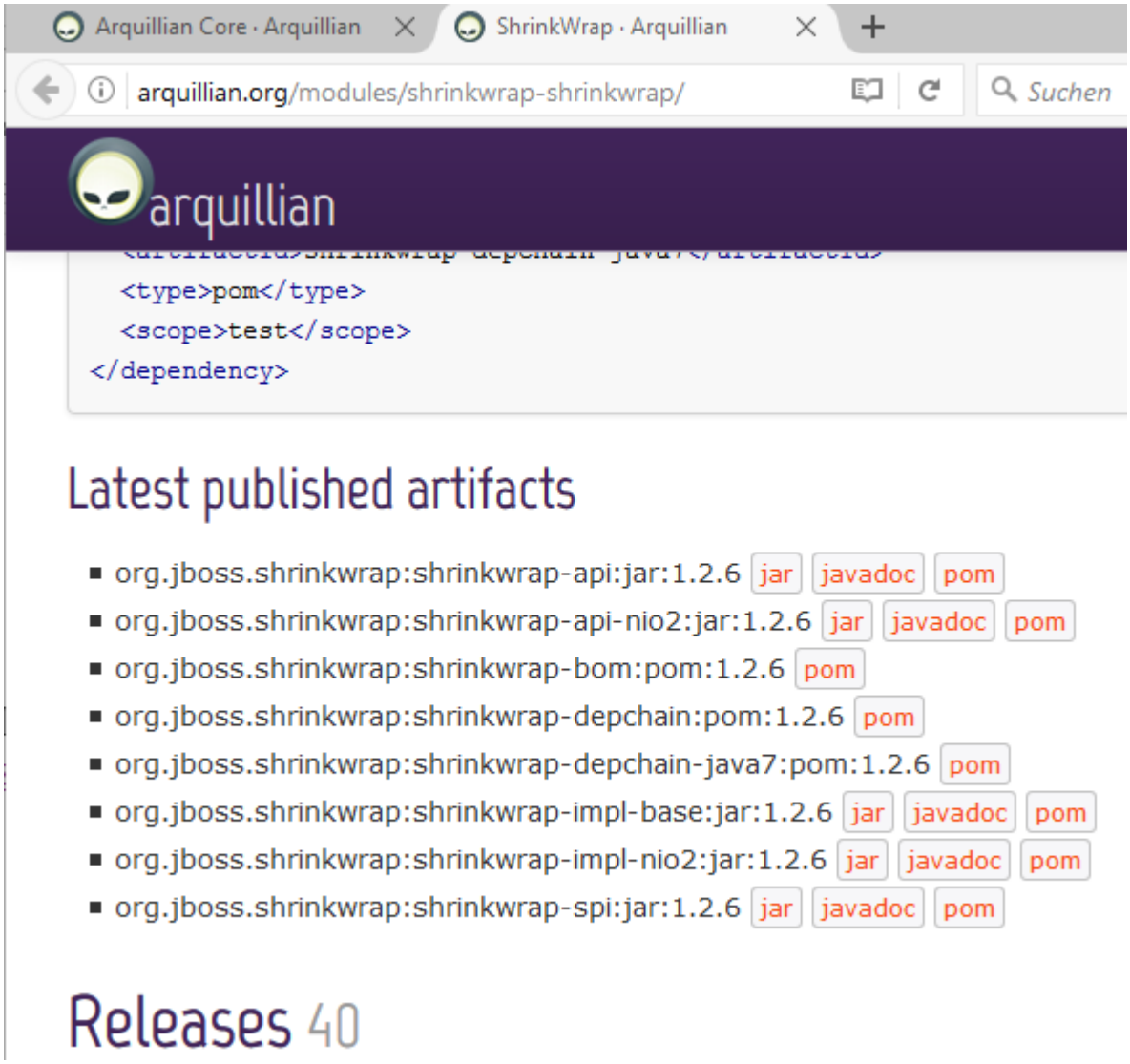
Generell sei angemerkt, dass das Aufsetzen von Arquillian in einer zur eigenen Projektumgebung passenden Version von vielen Personen als sehr aufwändig und frustrierend beschrieben wird. Nach dieser enormen Schwelle ist die eigentliche Nutzung dagegen extrem intuitiv. Um die benötigten Jar-Dateien zu erhalten, gibt es verschiedene Wege. Zwei davon sind in den beiden nachfolgenden Unterkapiteln beschrieben, von denen natürlich nur einer genutzt werden muss.

10.6.1 Herunterladen vom Hersteller

Da offiziell nur Build-Werkzeuge unterstützt werden, ist die manuelle Installation recht aufwändig, da viele jar-Dateien von Hand im Netz gesucht werden müssen. Einen Überblick über die benötigten Dateien ist mit älteren Versionsinformationen in https://gist.github.com/mojavelinux/2363038 beschrieben, die Seite http://arquillian.org/modules/core-platform/ enthält im Bereich „Latest published artifacts“ Links zu jar-Dateien. Dateien mit testng im Namen sowie junit-standalone werden beim Download weggelassen.



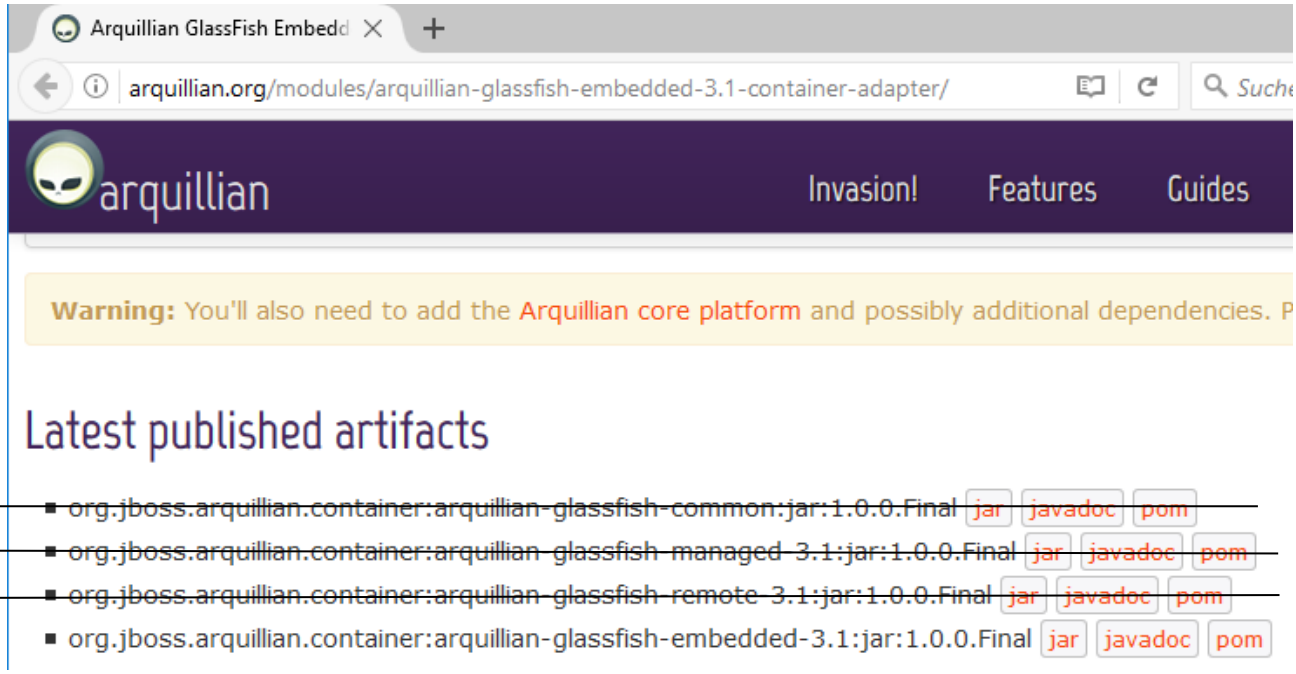
Die von ShrinkWrap benötigten Dateien sind auf der Seite http://arquillian.org/modules/shrinkwrap-shrinkwrap/ wieder unter „Latest published artifacts“ verlinkt.



Zu Shrinkwarap werden Descriptoren benötigt, die unter http://arquillian.org/modules/descriptors-shrinkwrap/ stehen. Die Jar-Dateien stehen wieder unter „Latest published artifacts“ verlinkt.

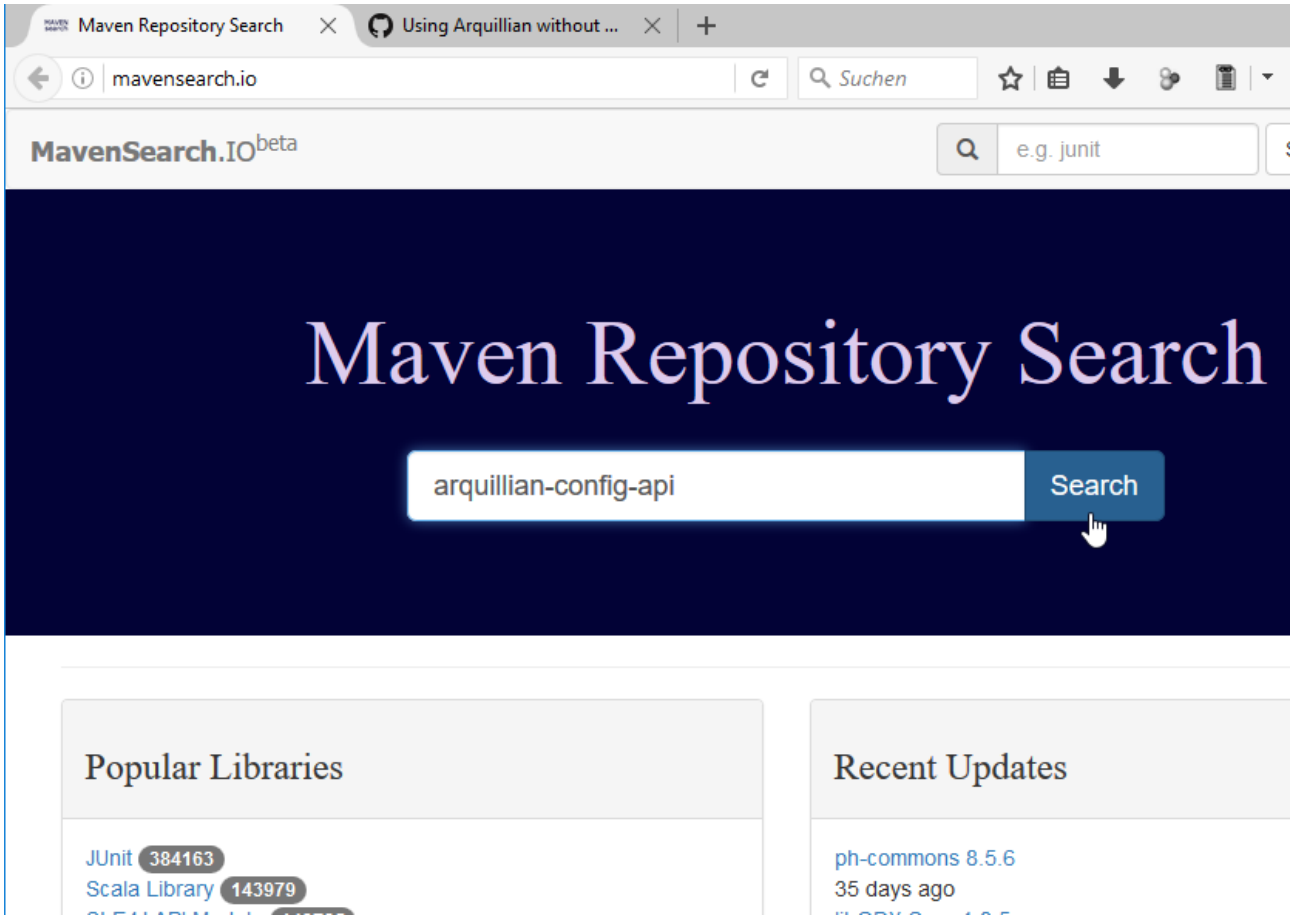


Weiterhin wird die Anbindung an den GlassFish benötigt, die unter http://arquillian.org/modules/arquillian-glassfish-embedded-3.1-container-adapter/ erhältlich ist. Die Jar-Dateien stehen wieder unter „Latest published artifacts“ verlinkt. Hier wird nur die Library zu arquillian-glassfish-embedded benötigt.

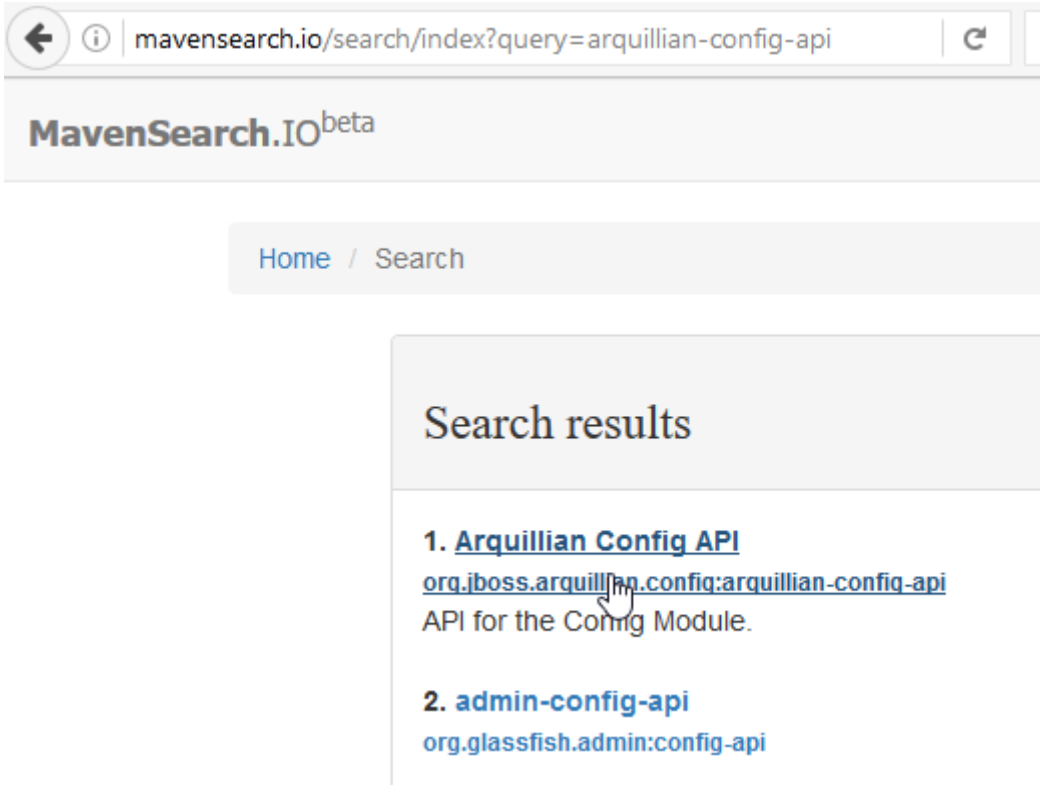


10.6.2 Herunterladen der von Maven genutzten Version

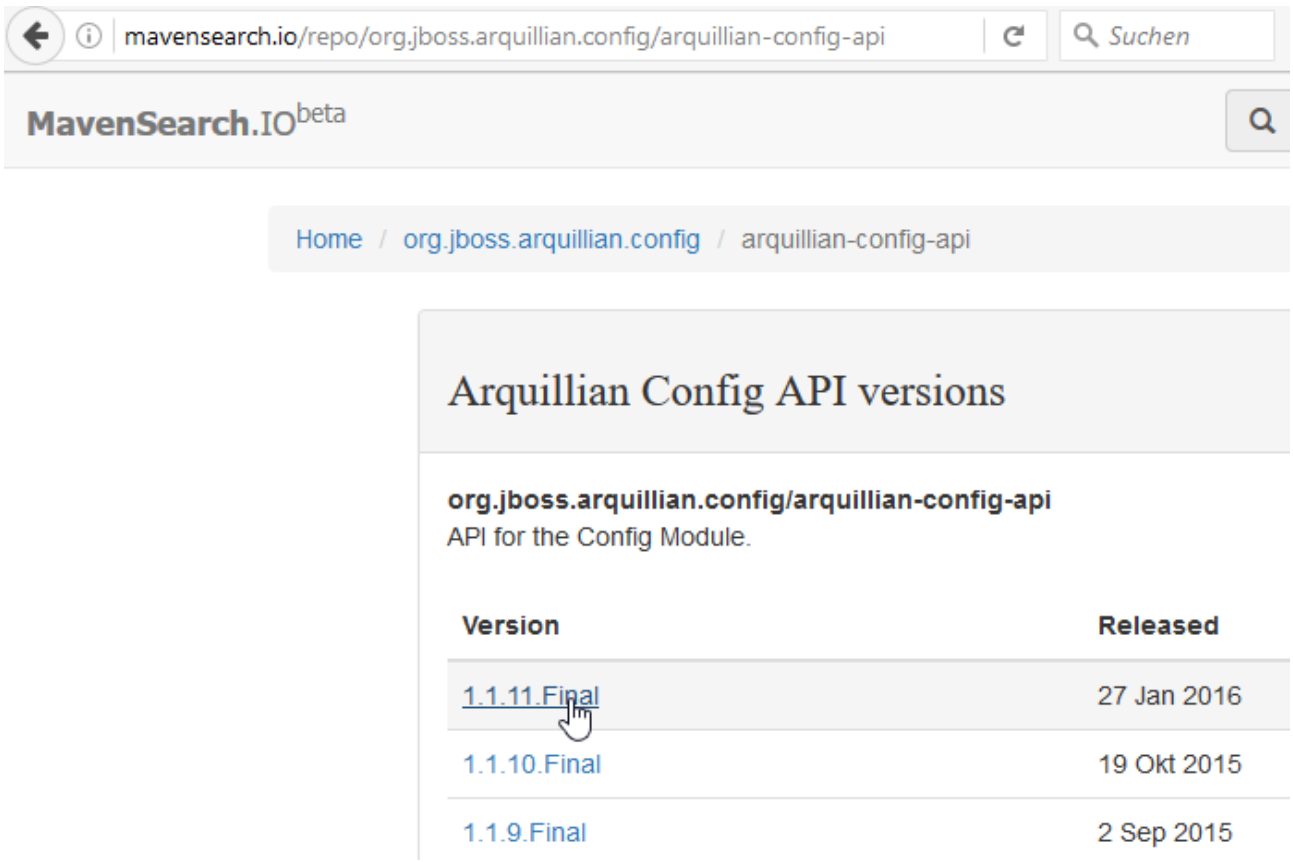
Die benötigten Dateien kann man alternativ direkt im Maven Repository von der Seite http://mavensearch.io/ herunterladen. Der Namen der gesuchten jar-Datei wird meist ohne Versionsnummer in das obere Suchfenster eingegeben.



Es wird des passende Treffer, meist der erste, erkennbar am passenden Paketnamen, angeklickt.



Es wird die neueste stabile Version angeklickt. Die Versionsnummern müssen später einheitlich sein.



Es wird dann die zugehötige jar-Datei über den Link heruntergeladen.Es kann sinnvoll sein, ebenfalls die jar-Datei mit javadoc-Inhalt herunterzuladen.



Die benötigten Dateien sind (hier nur Links zur Version 1.1.2):

- org.jboss.arquillian.core » arquillian-core-api jar javadoc
- org.jboss.arquillian.core » arquillian-core-spi jar javadoc
- org.jboss.arquillian.core » arquillian-core-impl-base jar javadoc
- org.jboss.arquillian.config » arquillian-config-api jar javadoc
- (hier nicht benötigt) org.jboss.arquillian.config » arquillian-config-spi jar javadoc
- org.jboss.arquillian.config » arquillian-config-impl-base jar javadoc
- org.jboss.arquillian.test » arquillian-test-api jar javadoc
- org.jboss.arquillian.test » arquillian-test-spi jar javadoc
- org.jboss.arquillian.test » arquillian-test-impl-base jar javadoc
- org.jboss.arquillian.container » arquillian-container-spi jar javadoc
- org.jboss.arquillian.container » arquillian-container-impl-base jar javadoc
- org.jboss.arquillian.container » arquillian-container-test-api jar javadoc

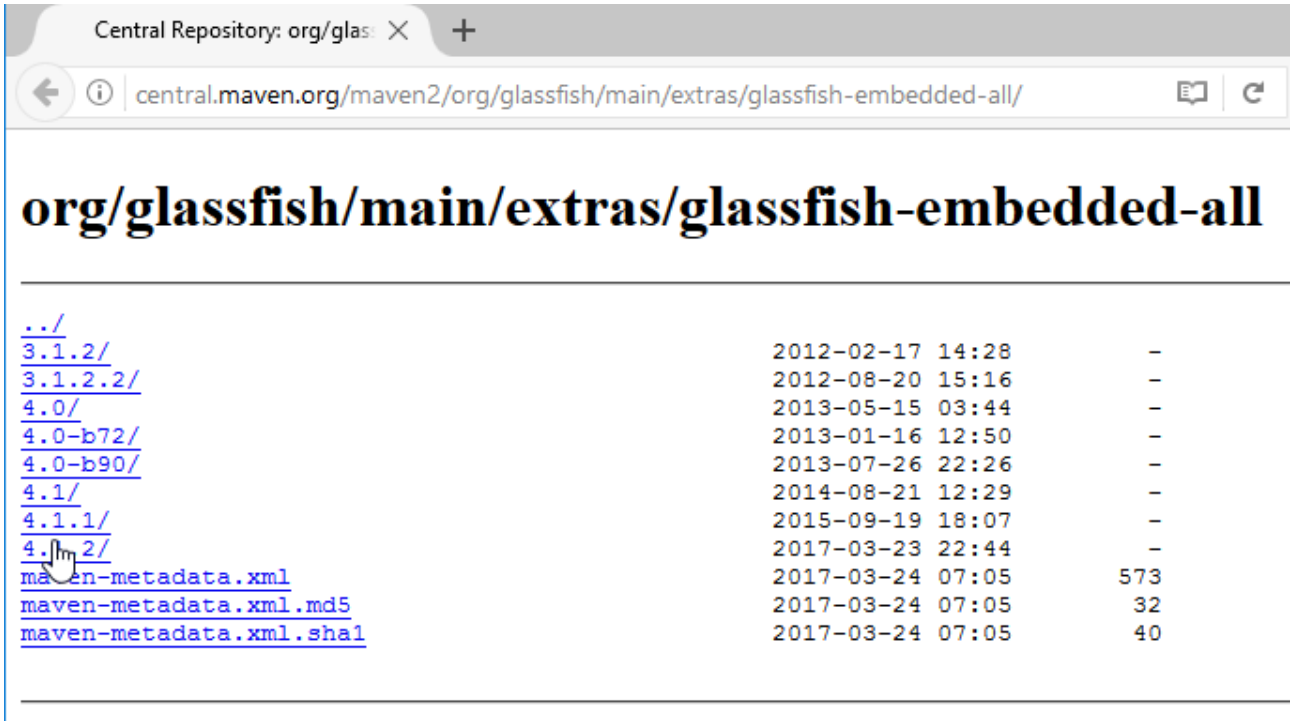
- org.jboss.arquillian.container » arquillian-container-test-spi jar javadoc
- org.jboss.arquillian.container » arquillian-container-test-impl-base jar javadoc
- org.jboss.arquillian.junit » arquillian-junit-core jar javadoc
- (nicht benötigt) org.jboss.arquillian.junit » arquillian-junit-standalone jar javadoc
- org.jboss.arquillian.junit » arquillian-junit-container jar javadoc
- org.jboss.arquillian.testenricher » arquillian-testenricher-cdi jar javadoc
- org.jboss.arquillian.testenricher » arquillian-testenricher-ejb jar javadoc
- org.jboss.arquillian.testenricher » arquillian-testenricher-resource jar javadoc
- org.jboss.arquillian.testenricher » arquillian-testenricher-initialcontext jar javadoc
- org.jboss.arquillian.protocol » arquillian-protocol-servlet jar javadoc
- (nicht benötigt) org.jboss.arquillian.protocol » arquillian-protocol-jmx jar javadoc

Weiterhin

- shrinkwrap-api-1.2.6.jar, http://central.maven.org/maven2/org/jboss/shrinkwrap/shrinkwrap-api/1.2.6/shrinkwrap-api-1.2.6.jar
- shrinkwrap-descriptors-api-base-2.0.0-alpha-10.jar, http://central.maven.org/maven2/org/jboss/shrinkwrap/descriptors/shrinkwrap-descriptors-api-base/2.0.0-alpha-10/shrinkwrap-descriptors-api-base-2.0.0-alpha-10.jar
- shrinkwrap-descriptors-impl-base-2.0.0-alpha-10.jar, http://central.maven.org/maven2/org/jboss/shrinkwrap/descriptors/shrinkwrap-descriptors-impl-base/2.0.0-alpha-10/shrinkwrap-descriptors-impl-base-2.0.0-alpha-10.jar
- shrinkwrap-descriptors-spi-2.0.0-alpha-10.jar, http://central.maven.org/maven2/org/jboss/shrinkwrap/descriptors/shrinkwrap-descriptors-spi/2.0.0-alpha-10/shrinkwrap-descriptors-spi-2.0.0-alpha-10.jar
- shrinkwrap-impl-base-1.2.6.jar, http://central.maven.org/maven2/org/jboss/shrinkwrap/shrinkwrap-impl-base/1.2.6/shrinkwrap-impl-base-1.2.6.jar
- shrinkwrap-spi-1.2.6.jar, http://central.maven.org/maven2/org/jboss/shrinkwrap/shrinkwrap-api/1.2.6/shrinkwrap-api-1.2.6.jar

10.6.3 Embedded Glassfish

Weiterhin benötigt wird der embedded Glassfish-Server glassfish-embedded-all-4.1.1.jar, http://central.maven.org/maven2/org/glassfish/main/extras/glassfish-embedded-all/4.1.1/glassfish-embedded-all-4.1.1.jar bzw. der neuste Glassfish mit ähnlichwer Endung aus dem Verzeichnis http://central.maven.org/maven2/org/glassfish/main/extras/glassfish-embedded-all/.

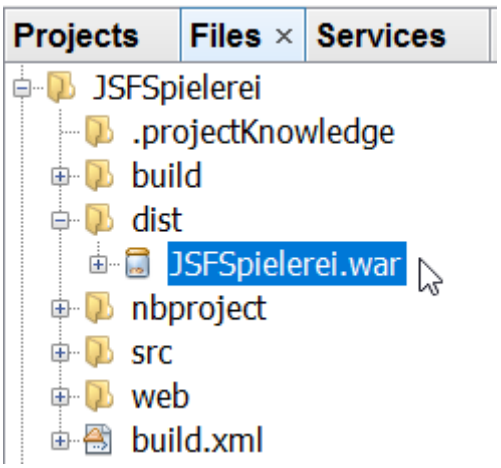


Für JUnit werden weiterhin junit-4.12.jar und hamcrest-core-1.3.jar benötigt.

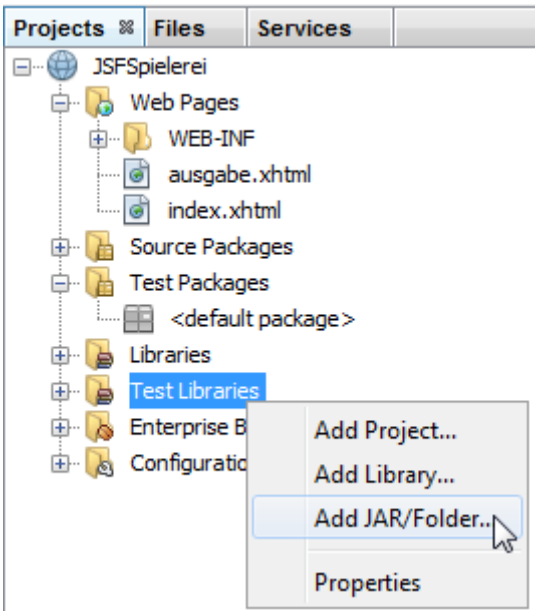
10.6.4 Beispielnutzung

Als Beispiel zum Testen wird das Projekt aus Abschnitt „10.4 Einbindung von JPA mit EJB-Hilfe“ genutzt.

Ein Blick auf die Dateien des Projekts zeigt, dass sich im Verzeichnis dist die Datei JSFSpielerei.war befindet, die auf dem Server eingespielt wird. Die Datei ist durch einen Klick auf „Clean & Build“ immer wieder neu erstellbar.

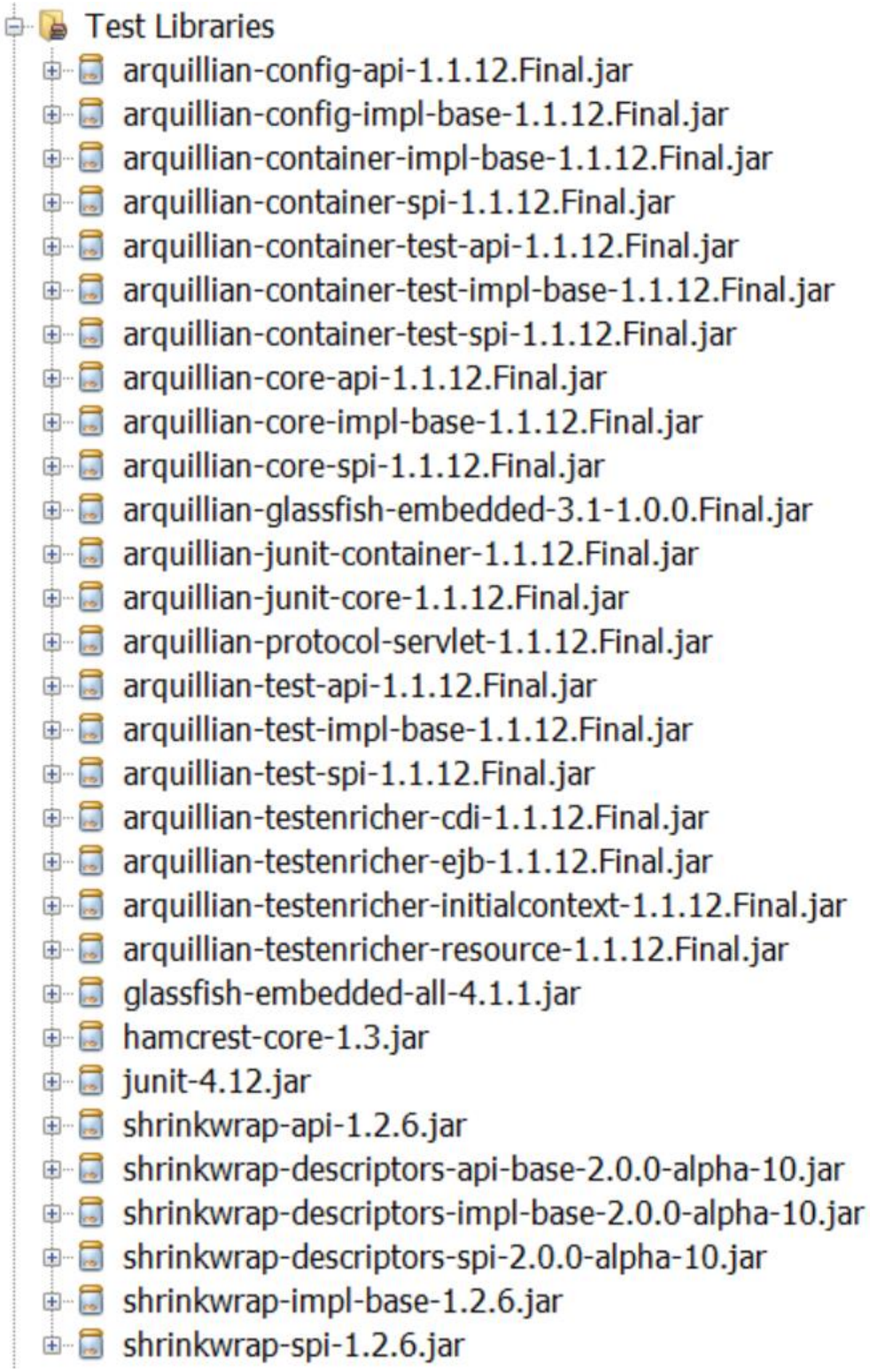


Nun werden alle jar-Dateien in einen Unterordner lib des Projekts kopiert und nach dem in „4.6 Nutzung von Bibliotheken/Jar-Files (NetBeans 8)“ beschriebenen Verfahren zu den „Test Libraries“ hinzugefügt.

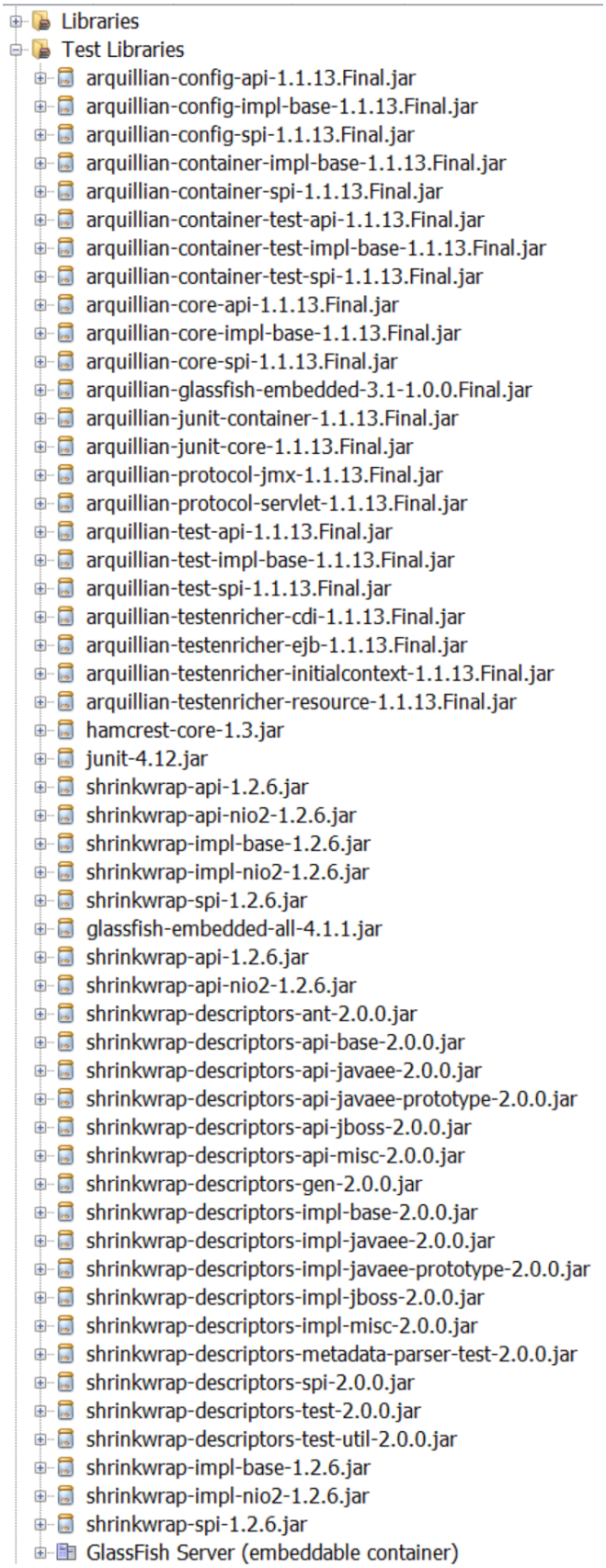


Die Einträge sollten wie folgt aussehen, wobei hier einige Libraries für TestNG entfernt wurden. Dieser Schritt ist nicht notwendig, es können einfach alle vorher „eingesammelten“ Libraries für Arquillian, ShrinkWrap, embedded Glassfish, JUnit und Hamcrest Core eingebunden werden.

.



Eine vollständige Einbindung führt zu folgendem Aufbau.


Libraries
Test Libraries
arquillian-config-api-1.1.13.Final.jar
arquillian-config-impl-base-1.1.13.Final.jar
arquillian-config-spi-1.1.13.Final.jar
arquillian-container-impl-base-1.1.13.Final.jar
arquillian-container-spi-1.1.13.Final.jar
arquillian-container-test-api-1.1.13.Final.jar
arquillian-container-test-impl-base-1.1.13.Final.jar
arquillian-container-test-spi-1.1.13.Final.jar
arquillian-core-api-1.1.13.Final.jar
arquillian-core-impl-base-1.1.13.Final.jar
arquillian-core-spi-1.1.13.Final.jar
arquillian-glassfish-embedded-3.1-1.0.0.Final.jar
arquillian-junit-container-1.1.13.Final.jar
arquillian-junit-core-1.1.13.Final.jar
arquillian-protocol-jmx-1.1.13.Final.jar
arquillian-protocol-servlet-1.1.13.Final.jar
arquillian-test-api-1.1.13.Final.jar
arquillian-test-impl-base-1.1.13.Final.jar
arquillian-test-spi-1.1.13.Final.jar
arquillian-testenricher-cdi-1.1.13.Final.jar
arquillian-testenricher-ejb-1.1.13.Final.jar
arquillian-testenricher-initialcontext-1.1.13.Final.jar
arquillian-testenricher-resource-1.1.13.Final.jar
hamcrest-core-1.3.jar
junit-4.12.jar
shrinkwrap-api-1.2.6.jar
shrinkwrap-api-nio2-1.2.6.jar
shrinkwrap-impl-base-1.2.6.jar
shrinkwrap-impl-nio2-1.2.6.jar
shrinkwrap-spi-1.2.6.jar
glassfish-embedded-all-4.1.1.jar
shrinkwrap-api-1.2.6.jar
shrinkwrap-api-nio2-1.2.6.jar
shrinkwrap-descriptors-ant-2.0.0.jar
shrinkwrap-descriptors-api-base-2.0.0.jar
shrinkwrap-descriptors-api-javaee-2.0.0.jar
shrinkwrap-descriptors-api-javaee-prototype-2.0.0.jar
shrinkwrap-descriptors-api-jboss-2.0.0.jar
shrinkwrap-descriptors-api-misc-2.0.0.jar
shrinkwrap-descriptors-gen-2.0.0.jar
shrinkwrap-descriptors-impl-base-2.0.0.jar
shrinkwrap-descriptors-impl-javaee-2.0.0.jar
shrinkwrap-descriptors-impl-javaee-prototype-2.0.0.jar
shrinkwrap-descriptors-impl-jboss-2.0.0.jar
shrinkwrap-descriptors-impl-misc-2.0.0.jar
shrinkwrap-descriptors-metadata-parser-test-2.0.0.jar
shrinkwrap-descriptors-spi-2.0.0.jar
shrinkwrap-descriptors-test-2.0.0.jar
shrinkwrap-descriptors-test-util-2.0.0.jar
shrinkwrap-impl-base-1.2.6.jar
shrinkwrap-impl-nio2-1.2.6.jar
shrinkwrap-spi-1.2.6.jar
GlassFish Server (embeddable container)

Sollen Änderungen am zu testenden Code vorgenommen werden, ist z. B. über „Clean&Build“ eine neue war-Datei zu erstellen.

Arquillian hat einen speziellen TestRunner, der unter anderem nach einem Server sucht, hier dem embedded GlassFish und diesen nutzt. Es wird eine speziell mit @Deployment annotierte Methode benötigt, mit der die war-Datei in den Server geladen wird. Danach können alle Klassen mit den bekannten Annotationen @RequestScoped, @Inject, @Stateless in den Tests zur Erstellung der zu testenden Objekte genutzt werden, was folgender Beispielcode zeigt. *Die zugehörige Datenbank muss erst von Hand gestartet werden. Weiterhin ist der zur Entwicklung genutzte GlassFish-Server unbedingt zu stoppen*, da der embedded GlassFish die gleichen Ports nutzt.

```
package test;

import controller.ModulBean;
import ejb.Modulverwaltung;
import entities.Modul;
import java.io.File;
import java.util.Date;
import java.util.List;
import javax.inject.Inject;
import org.jboss.arquillian.container.test.api.Deployment;
import org.jboss.arquillian.junit.Arquillian;
import org.jboss.shrinkwrap.api.ShrinkWrap;
import org.jboss.shrinkwrap.api.spec.WebArchive;
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(Arquillian.class)
public class BeispielTest {

    @Inject
    Modul modul;

    @Inject
    Modulverwaltung vw;

    @Inject
    ModulBean controller;

    @Deployment
    public static WebArchive createDeployment() {
        // hier ist der Pfad zur am Anfang des Kapitels genannten war-Datei wichtig
        WebArchive war = ShrinkWrap.createFromZipFile(WebArchive.class,
                 new File("dist/JSFSpielerei.war"));
        System.out.println("war: " + war + " " + war.getContent());
        return war;
    }

    @Test
    public void modulbasicsTest(){ //auch als einfaches JUnit moeglich
```

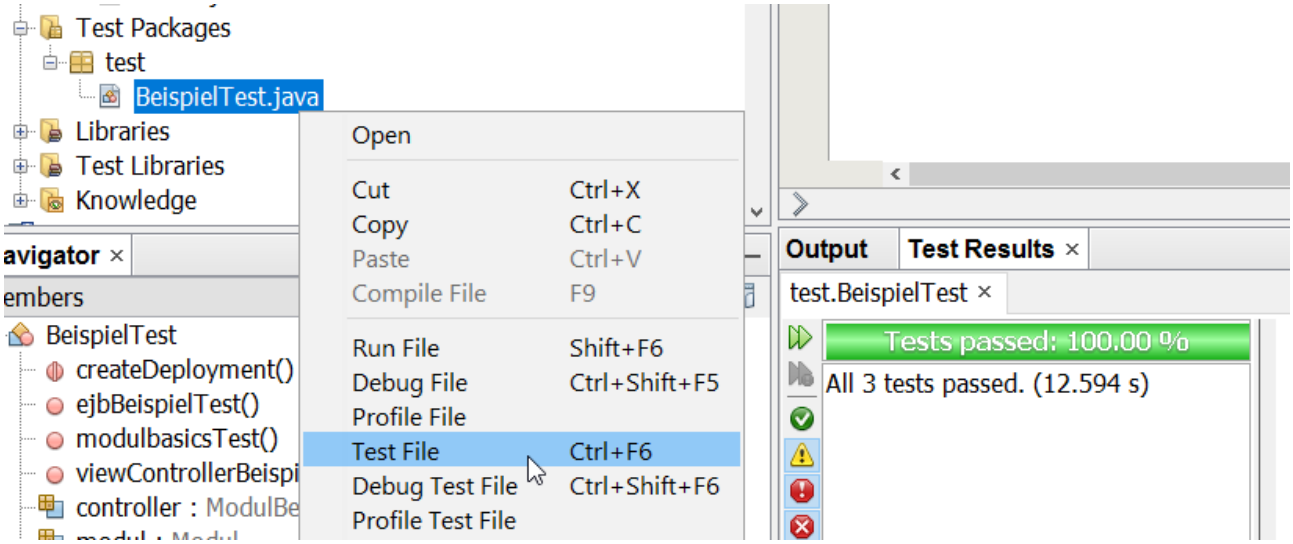
```
            Assert.assertNotNull("Modul nicht erzeugt", modul);
            this.modul.setName("Blubb");
            this.modul.setNr(42);
            Assert.assertEquals("Blubb", this.modul.getName());
            Assert.assertEquals(42, this.modul.getNr());
        }

        @Test
        public void ejbBeispielTest(){
            List<Modul> alt = this.vw.getModule();
            String name = new Date().toString();
            this.modul.setName(name);
            this.vw.persist(this.modul);
            List<Modul> neu = this.vw.getModule();
            neu.removeAll(alt);
            Assert.assertEquals("ein neues Objekt erwartet", 1, neu.size());
            Assert.assertEquals("falscher Modulname"
                    , name, neu.get(0).getName());
        }

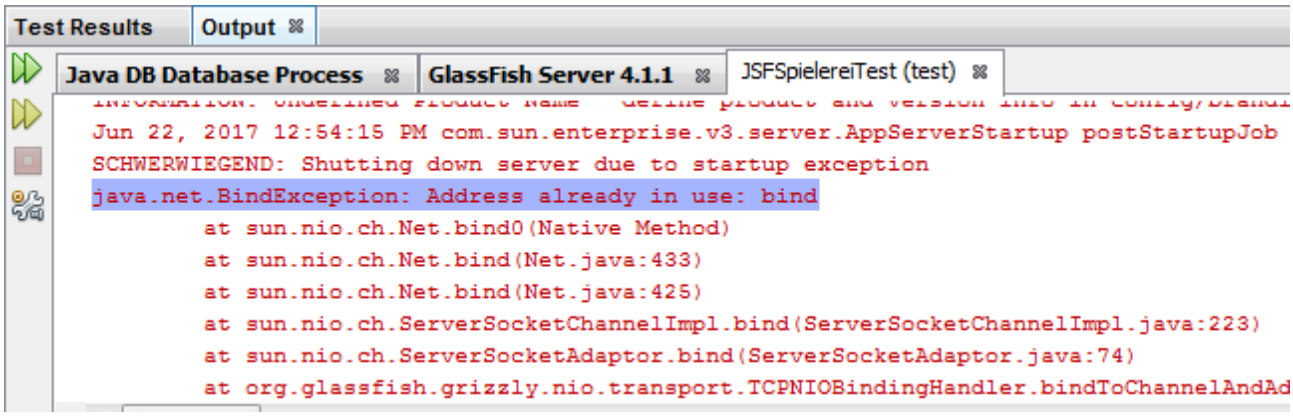
        @Test
        public void viewControllerBeispielTest(){
            List<Modul> alt = this.controller.getModule();
            String name = new Date().toString();
            this.modul.setName(name);
            this.controller.setModul(this.modul);
            this.controller.uebernehmen();
            List<Modul> neu = this.controller.getModule();
            neu.removeAll(alt);
            Assert.assertEquals("ein neues Objekt erwartet", 1, neu.size());
            Assert.assertEquals("falscher Modulname"
                    , name, neu.get(0).getName());
        }
    }
```

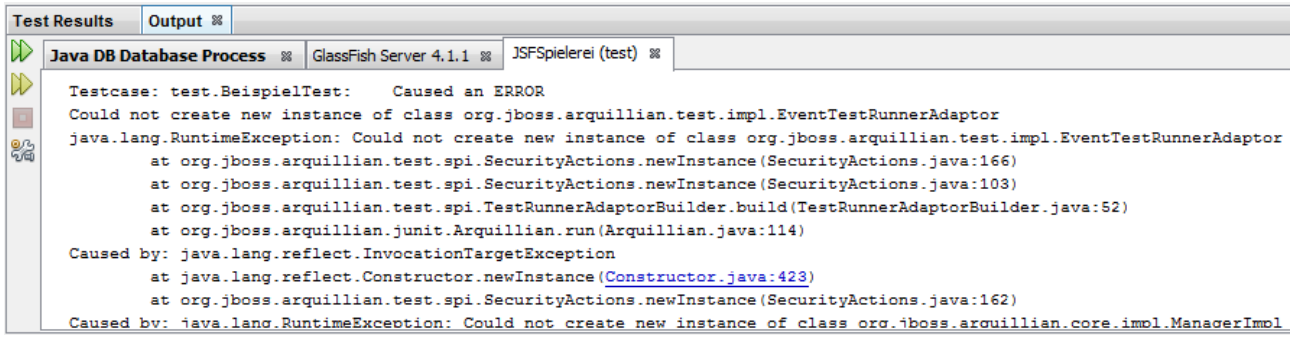
Die Testklasse zeigt die Möglichkeiten, nicht die Systematik beim Testen. Natürlich ist die Aufteilung in Tests pro Klasse sinnvoll. Ein direkter Zugriff auf die Datenbank ist auch grundsätzlich denkbar. Weiterhin ist leicht ersichtlich, dass nicht alle Methoden optimal testbar sind und dass bei der Erstellung von Programmen schon über ihre Testbarkeit nachgedacht werden muss.

Die Tests werden wie normale JUnit-Tests ausgeführt.



Sollte der ursprüngliche GlassFish-Server noch laufen, wird im Output auf „Adress already in use“ hingewiesen. Alternativ ist auch eine RuntimeException möglich.





10.7 Sichere JSF-Applikationen

Generell müssen für den Bereich Security zwei Teilbereiche betrachtet werden. Es gibt den JSF-spezifischen Anteil, mit dem festgelegt werden kann, welche Rollen auf welche Seiten zugreifen

können. Weiterhin gibt es einen Server-spezifischen Anteil, der die Zugänge der Nutzer und deren Zuordnung zu Gruppen regelt. Diese Gruppen oder auch einzelne Nutzer können dann den JSF-Rollen zugeordnet werden.

In diesem Abschnitt werden die Nutzer in einer getrennten Datenbank verwaltet, die über JDBC eingebunden wird. Eine direkte Einrichtung von Nutzern in GlassFish wäre auch möglich und ist in „8.6 Sichere JSP-Seiten“ beschrieben.

10.7.1 Einrichten der Berechtigungstabellen

Die Passwörter sollen MD5-kodiert sein, es werden später die Nutzer edna, uwe und otto mit den Nutzernamen als Passwort gespeichert. Die Berechnung der Kodierung erfolgt durch folgendes Programm.

```
package md5;

import java.io.UnsupportedEncodingException;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Main {

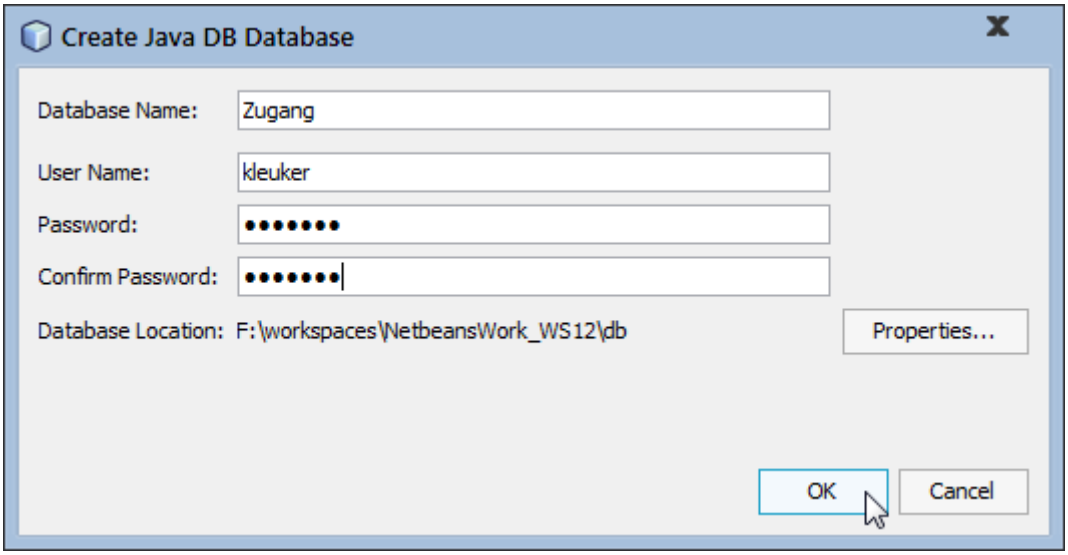
  public static void main(String[] args) throws NoSuchAlgorithmException
          , UnsupportedEncodingException {
    String[] nutzer = {"edna", "uwe", "otto"};
    MessageDigest md = MessageDigest.getInstance("MD5");
    for(String s:nutzer){
      byte[] digest = md.digest(s.getBytes("UTF-8"));
      System.out.println(s+" pw:"+new BigInteger(1,digest).toString(16));
    }
  }
}
```

Die Ausgabe lautet.

```
edna pw:f5fa02bc60f633f3b1781a824f5211b5
uwe pw:c418f34f261efe1473465ade95bfc22c
otto pw:e5645dd85deb100fd1d71d0e8d671091
```

Obwohl die Strings 32 Zeichen lang sind, bleiben kurze Passwörter ein enormes Risiko, da man durch einfaches googeln zu den Strings hier die realen kurzen Passwörter bestimmen kann.

Es wird eine JavaDB-Datenbank mit beliebigen Namen, hier „Zugang“, wie in „6.3 Anlegen einer Tabelle“ beschrieben, mit Use Name und Passwort „kleuker“ angelegt. Generell ist zu beachten, dass man sich immer genau an die gewählte Schreibweise hält, da man fast nie gute Fehlermeldungen enthält. Oftmals ist es die einzige Chance die gesamten Schritte erneut durchzuführen.



Weiterhin werden zwei Tabellen ergänzt, was wie in „**Fehler! Verweisquelle konnte nicht gefunden werden.Fehler! Verweisquelle konnte nicht gefunden werden.**“ beschrieben, passieren kann, und mit Beispieldaten gefüllt.

```
CREATE TABLE Nutzer(
  username VARCHAR(30),
  password VARCHAR(100),
  PRIMARY KEY (username)
);

INSERT INTO Nutzer VALUES('edna','f5fa02bc60f633f3b1781a824f5211b5');
INSERT INTO Nutzer VALUES('uwe','c418f34f261efe1473465ade95bfc22c');
INSERT INTO Nutzer VALUES('otto','e5645dd85deb100fd1d71d0e8d671091');

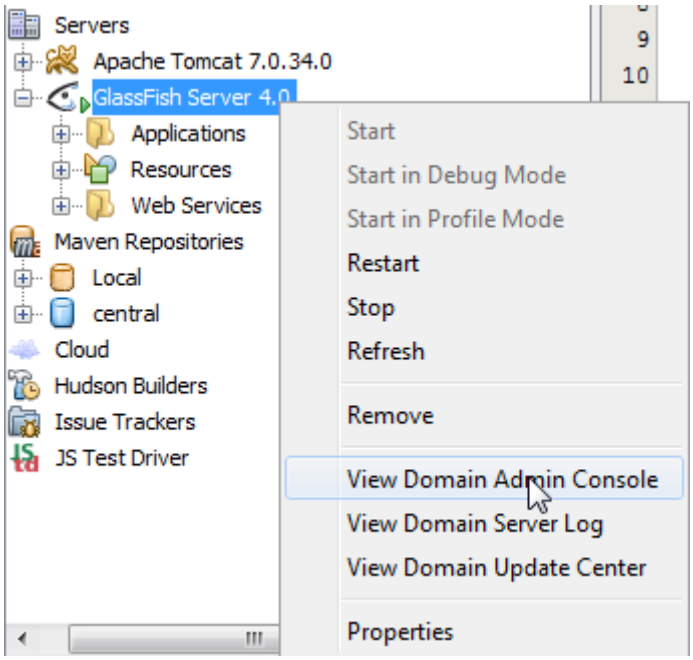
CREATE TABLE Gruppen(
  groupid VARCHAR(20),
  username VARCHAR(30),
  PRIMARY KEY (groupid,username),
  FOREIGN KEY (username) REFERENCES Nutzer(username)
);

INSERT INTO Gruppen VALUES ('admin','edna');
INSERT INTO Gruppen VALUES ('basic','edna');
INSERT INTO Gruppen VALUES ('basic','uwe');
```

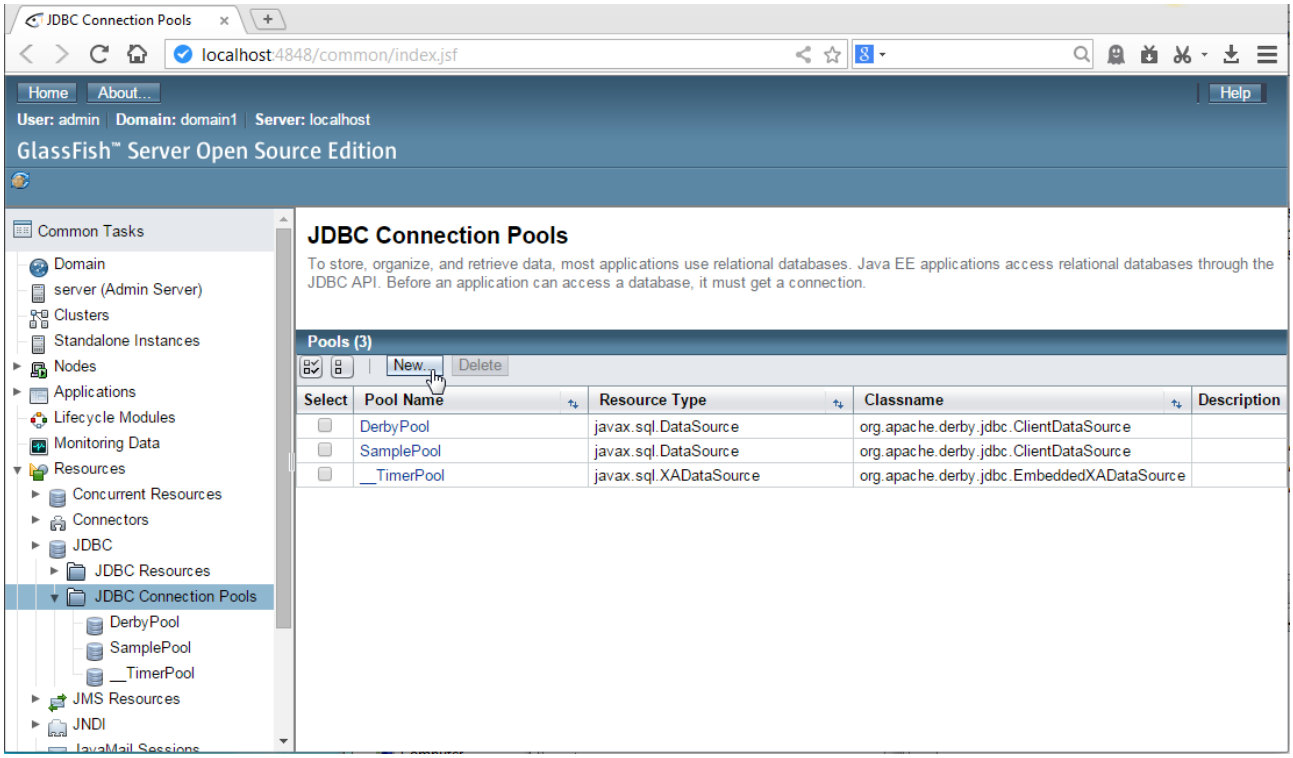
10.7.2 Datenbank in NetBeans bekannt machen

Im nächsten Schritt wird GlassFish so konfiguriert, dass die obige Datenbank bei Nutzerüberprüfungen genutzt wird. Dazu wird ein Rechtklick auf dem laufenden GlassFish-Server

gemacht und „View Domain Admin Console“ gewählt.

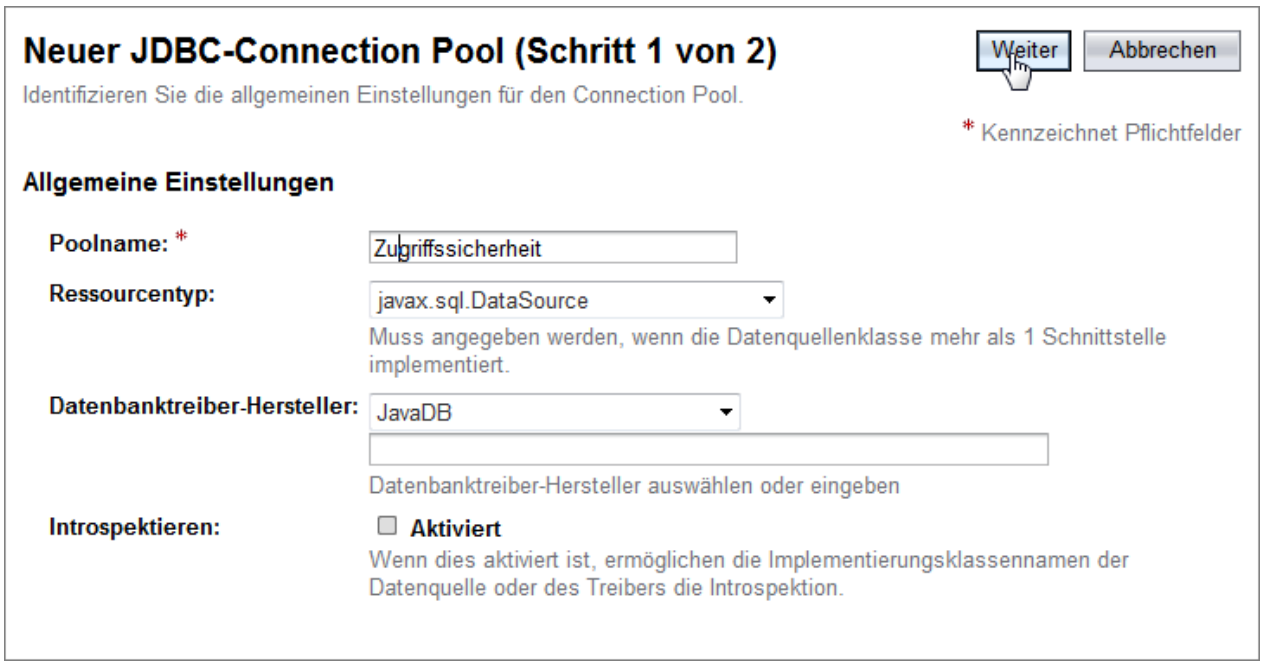


Man klappt links das Feld „JDBC“ auf, wählt „JDBC-Connection-Pools“ und klickt „Neu…“.

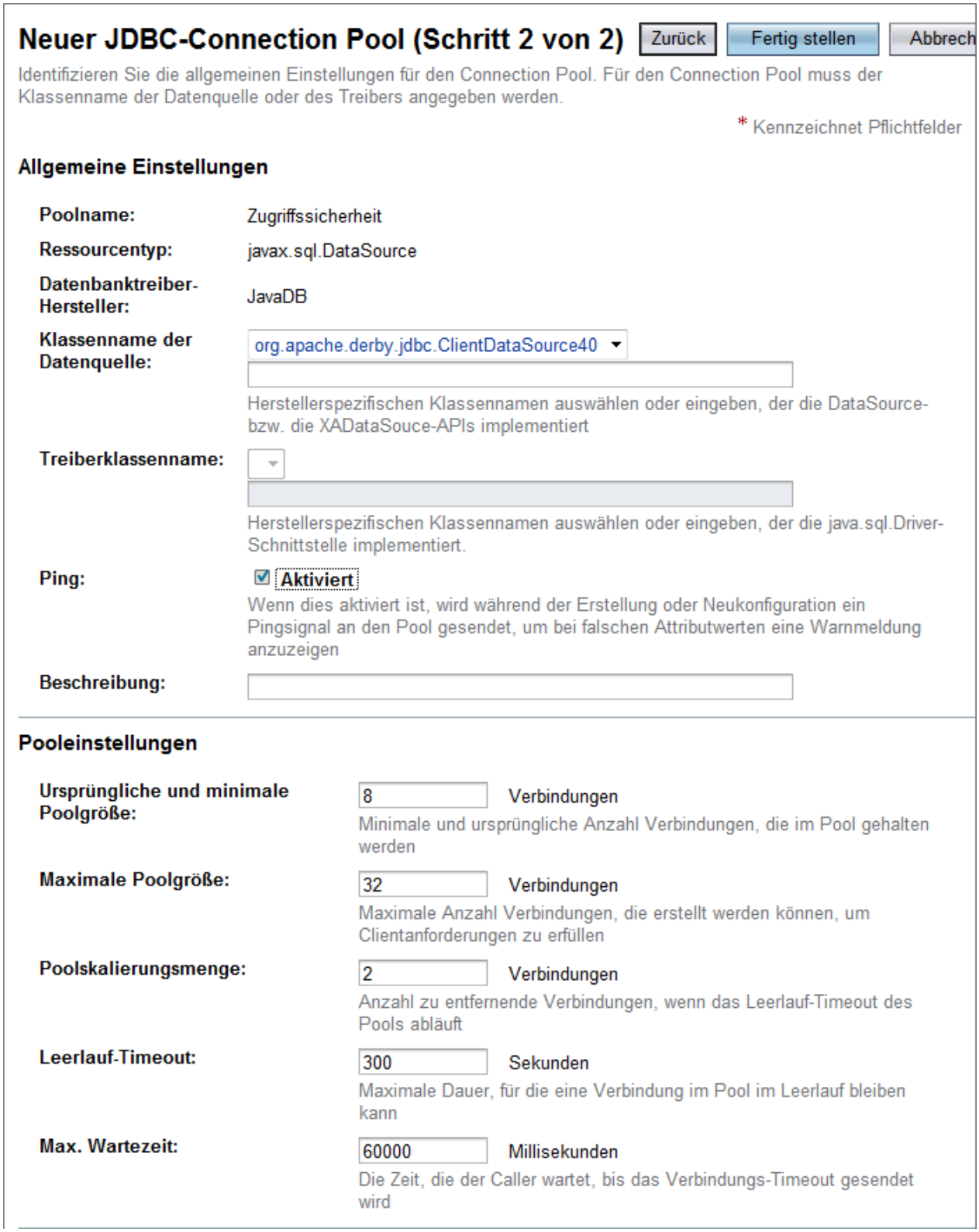


Man vergibt einen sinnvollen Poolnamen, hier „Zugriffssicherheit“, wählt als Ressourcentyp

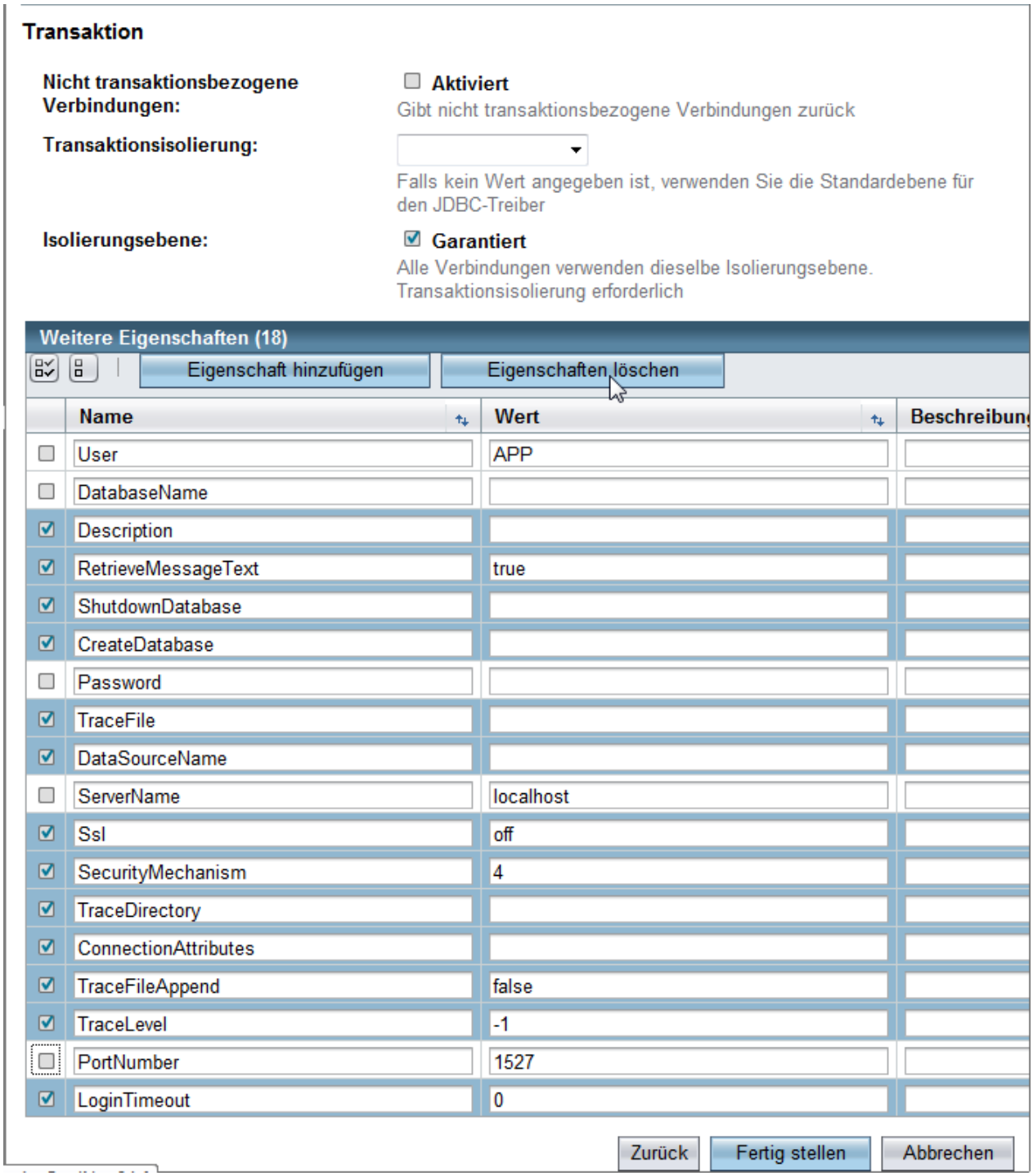
„javax.sql.DataSource“ und als Datenbanktreiber-Hersteller JavaDB aus und klickt rechts-oben auf „Weiter“.



Im oberen Bereich des Fensters ist es sinnvoll, einen Ping zu erlauben, um später prüfen zu können, ob die Datenbank erreichbar ist.


Neuer JDBC-Connection Pool (Schritt 2 von 2)
Zurück
Fertig stellen
Abbrech
Identifizieren Sie die allgemeinen Einstellungen für den Connection Pool. Für den Connection Pool muss der Klassenname der Datenquelle oder des Treibers angegeben werden.
* Kennzeichnet Pflichtfelder
Allgemeine Einstellungen
Poolname:
Zugriffssicherheit
Ressourcentyp:
javax.sql.DataSource
Datenbanktreiber-Hersteller:
JavaDB
Klassenname der Datenquelle:
org.apache.derby.jdbc.ClientDataSource40
Herstellerspezifischen Klassennamen auswählen oder eingeben, der die DataSource- bzw. die XADataSouce-APIs implementiert
Treiberklassenname:
Herstellerspezifischen Klassennamen auswählen oder eingeben, der die java.sql.Driver-Schnittstelle implementiert.
Ping:
Aktiviert
Wenn dies aktiviert ist, wird während der Erstellung oder Neukonfiguration ein Pingsignal an den Pool gesendet, um bei falschen Attributwerten eine Warnmeldung anzuzeigen
Beschreibung:
Pooleinstellungen
Ursprüngliche und minimale Poolgröße:
8
Verbindungen
Minimale und ursprüngliche Anzahl Verbindungen, die im Pool gehalten werden
Maximale Poolgröße:
32
Verbindungen
Maximale Anzahl Verbindungen, die erstellt werden können, um Clientanforderungen zu erfüllen
Poolskalierungsmenge:
2
Verbindungen
Anzahl zu entfernende Verbindungen, wenn das Leerlauf-Timeout des Pools abläuft
Leerlauf-Timeout:
300
Sekunden
Maximale Dauer, für die eine Verbindung im Pool im Leerlauf bleiben kann
Max. Wartezeit:
60000
Millisekunden
Die Zeit, die der Caller wartet, bis das Verbindungs-Timeout gesendet wird

Danach scrollt man weiter nach unten, wählt die in der folgenden Abbildung angezeigten Eigenschaften aus und löscht diese mit einem Klick auf „Eigenschaften löschen“. Die Zeilen ServerName und Ssl könnten vertauscht sein.

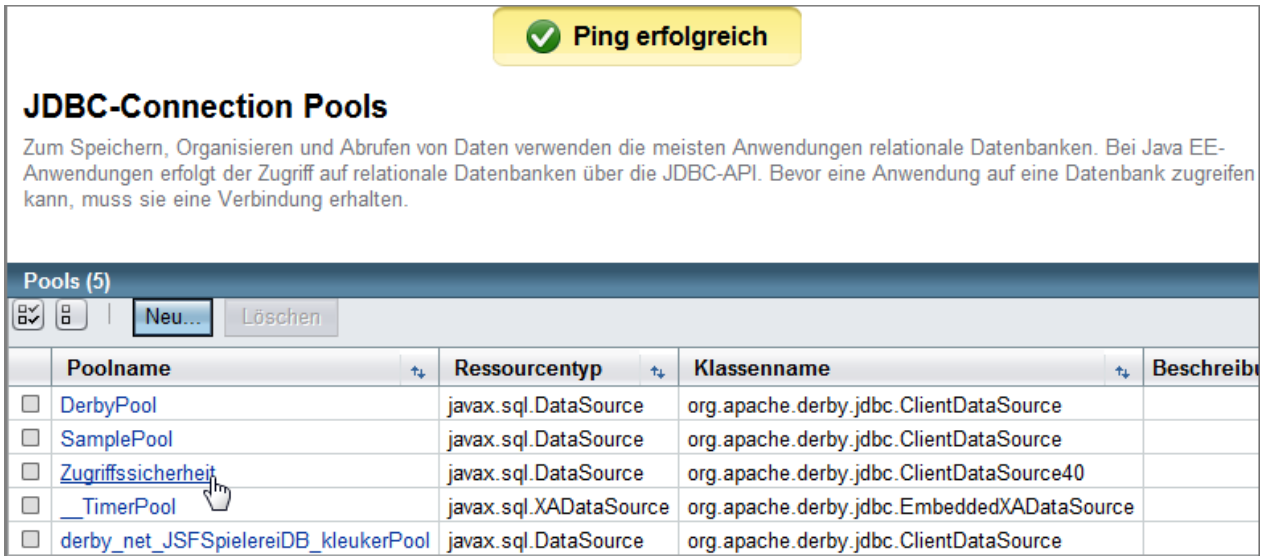


Die verbleibenden Eigenschaften werden mit den zur Datenbank passenden Werten aktualisiert und

auf „Fertig stellen“ geklickt.



Man sieht, dass die Datenbank erfolgreich mit GlassFish vernetzt wurde.



10.7.3 Datenbank über JNDI zugängig machen

Man klickt dann links auf „JDBC Ressourcen“ und dann auf „Neu…“.



Der JNDI Name ist theoretisch frei wählbar, sollte aber mit „jdbc/“ beginnen, um bereits am Namen als JDBC-Quelle sichtbar zu werden. Als Pool Name wird dann aus der Auswahl „Zugriffssicherheit“ gewählt und rechts-oben auf „OK“ geklickt. Die bisherigen Schritte können auch zur Einbindung einer beliebigen anderen Datenbank genutzt werden, wobei dann ein anderer Pool ausgewählt würde.

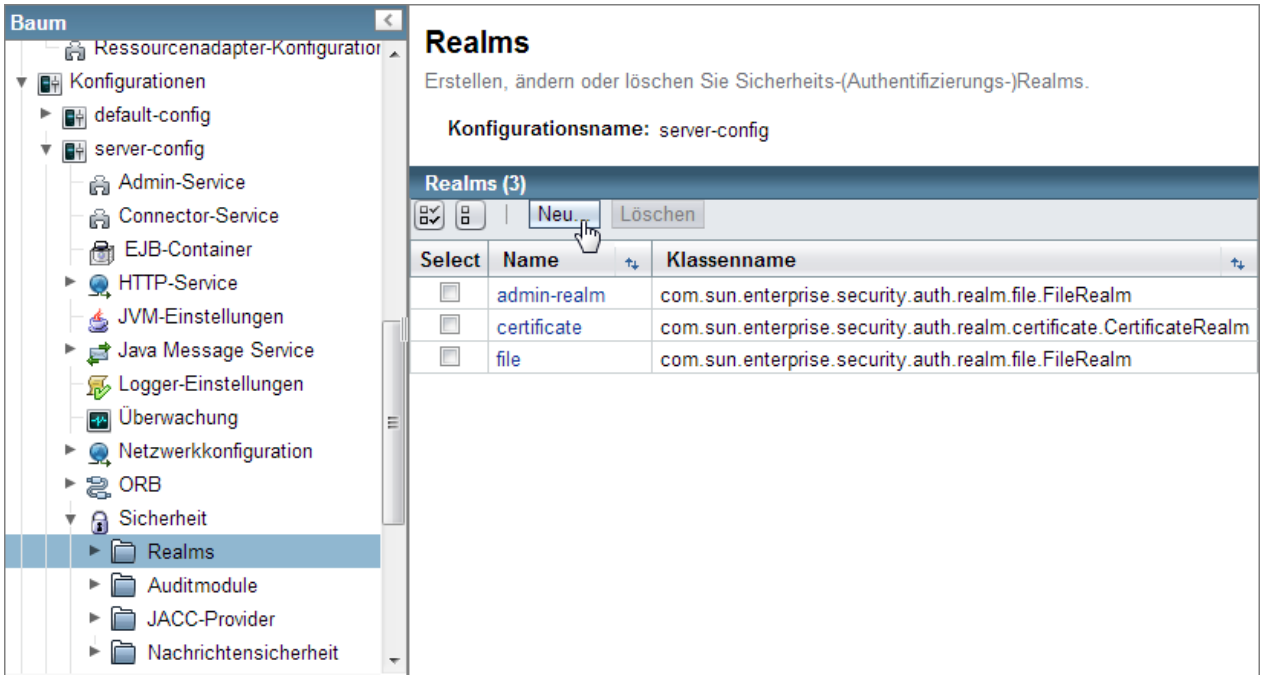


Man erkennt, dass die Ressource eingetragen wurde.

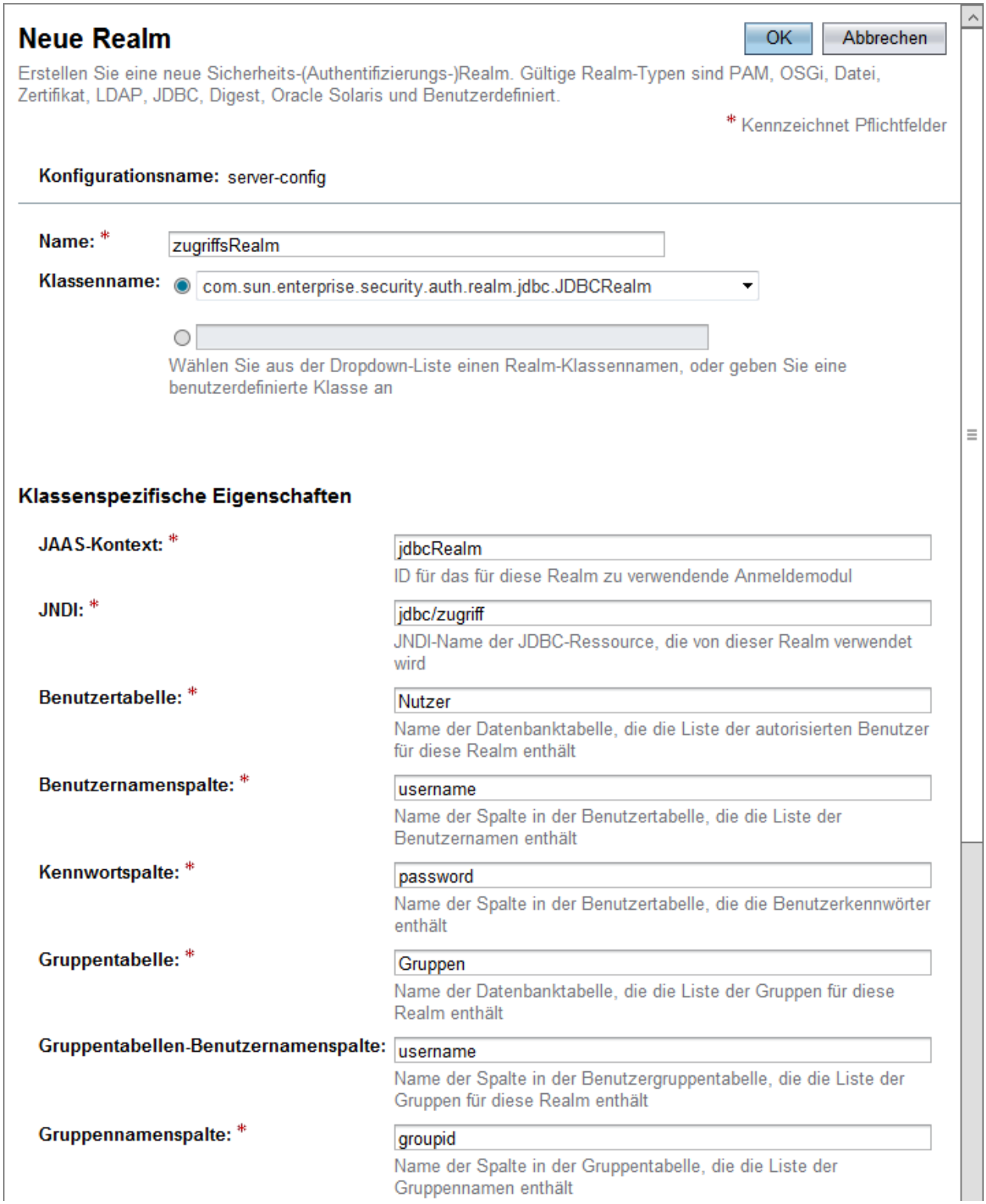


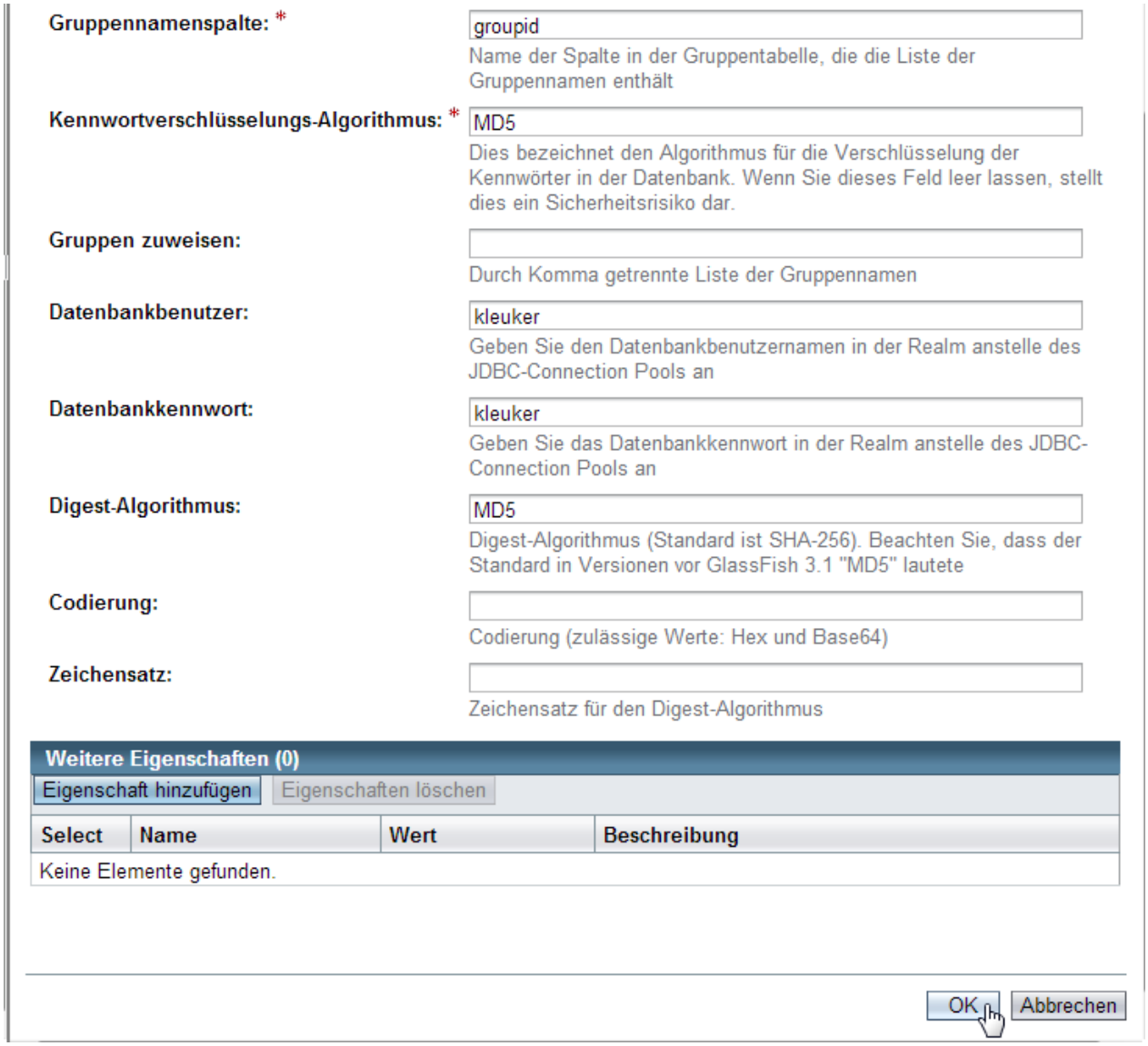
10.7.4 Eigenen Realm anlegen

Dann wird am linken Rand „server-config -> Sicherheit“ geöffnet, „Realms“ ausgewählt und „Neu…“ geklickt. (Das Wort realm, hier als Bereich übersetzbar, wird eingedeutscht „rälm“ ausgesprochen und hat damit nichts mit der Aussprache von „real“ zu tun.)



Bis auf den frei wählbaren Namen, hier „zugriffsRealm“, ergeben sich alle Einträge in den folgenden zwei Abbildungen vollständig aus den bisherigen Eintragungen. Man beachte, dass Tippfehler, die hier auftreten, extrem schwer später zu finden sind, da man meist nur mit Fehlermeldungen der Form „es ist ein Fehler aufgetreten“ konfrontiert wird. Es muss z.B. auch der Klassenname in der zweiten Zeile wie gezeigt geändert werden. Weiterhin ist die ID des JAAS-Kontexts, hier „jdbcRealm“, frei wählbar.


Neue Realm
OK
Abbrechen
Erstellen Sie eine neue Sicherheits-(Authentifizierungs-)Realm. Gültige Realm-Typen sind PAM, OSGi, Datei, Zertifikat, LDAP, JDBC, Digest, Oracle Solaris und Benutzerdefiniert.
* Kennzeichnet Pflichtfelder
Konfigurationsname: server-config
Name: *
zugriffsRealm
Klassenname:
com.sun.enterprise.security.auth.realm.jdbc.JDBCRealm
Wählen Sie aus der Dropdown-Liste einen Realm-Klassennamen, oder geben Sie eine benutzerdefinierte Klasse an
Klassenspezifische Eigenschaften
JAAS-Kontext: *
jdbcRealm
ID für das für diese Realm zu verwendende Anmeldemodul
JNDI: *
jdbc/zugriff
JNDI-Name der JDBC-Ressource, die von dieser Realm verwendet wird
Benutzertabelle: *
Nutzer
Name der Datenbanktabelle, die die Liste der autorisierten Benutzer für diese Realm enthält
Benutzernamenspalte: *
username
Name der Spalte in der Benutzertabelle, die die Liste der Benutzernamen enthält
Kennwortspalte: *
password
Name der Spalte in der Benutzertabelle, die die Benutzerkennwörter enthält
Gruppentabelle: *
Gruppen
Name der Datenbanktabelle, die die Liste der Gruppen für diese Realm enthält
Gruppentabellen-Benutzernamenspalte:
username
Name der Spalte in der Benutzergruppentabelle, die die Liste der Gruppen für diese Realm enthält
Gruppennamenspalte: *
groupid
Name der Spalte in der Gruppentabelle, die die Liste der Gruppennamen enthält

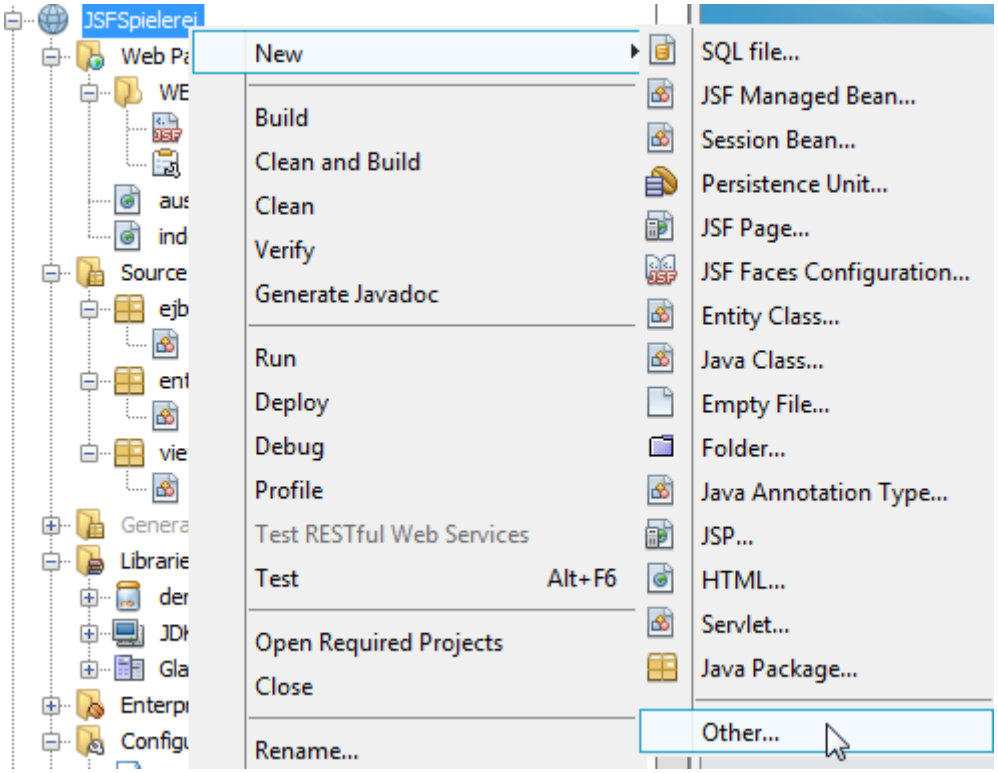


Man erkennt, dass der Realm eingetragen wurde. (In der Version 4.0.1b05 ist der existierende Eintrag leider nicht sichtbar, hier ist es notwendig.)

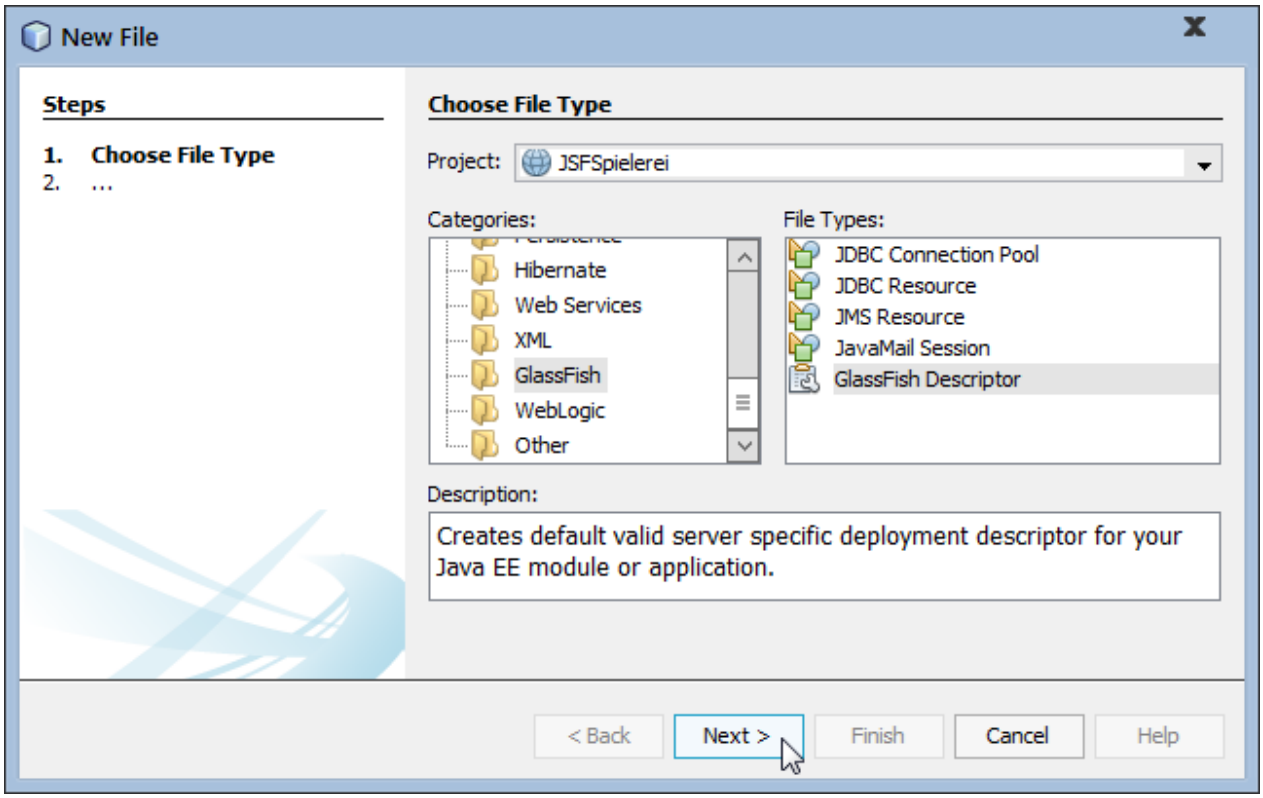


10.7.5 GlassFish-spezifische Einstellungen

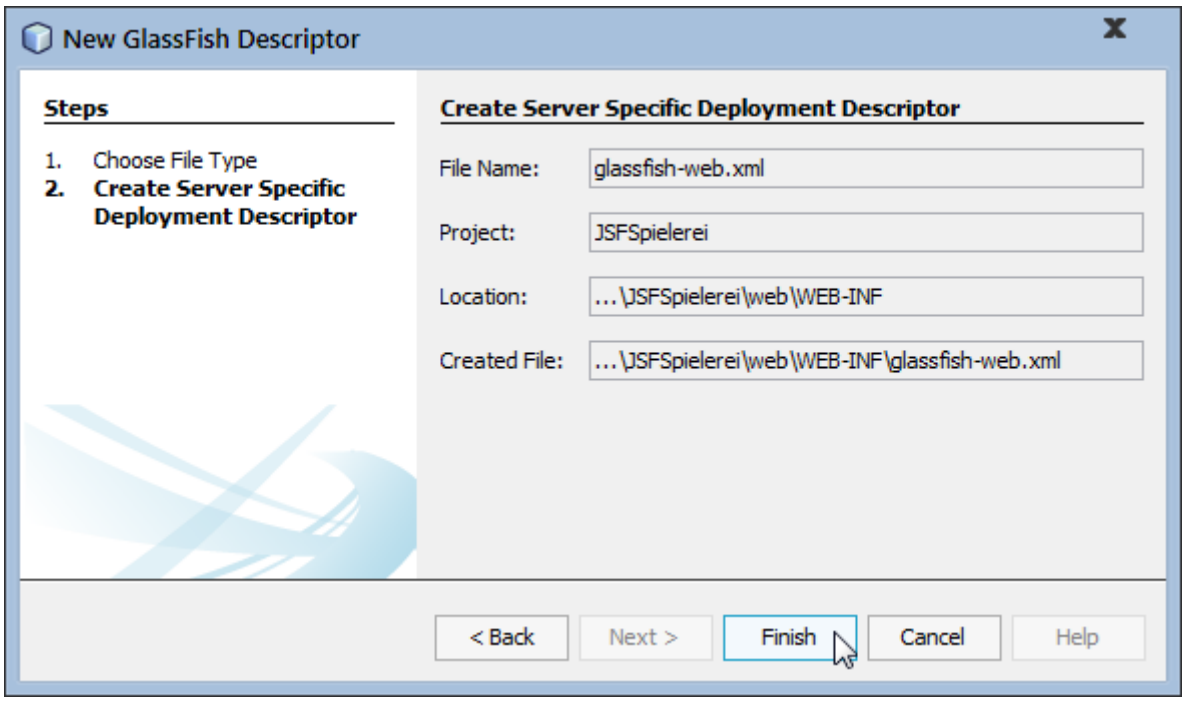
Nun soll die bisherige Applikation so eingerichtet werden, dass nur Nutzer der Gruppe „basic“ neue Module eintragen können. Dazu wird zunächst die Verbindung der Applikation mit der Nutzerverwaltung eingerichtet. Dies passiert in einer GlassFish-spezifischen Datei glassfisch-web.xml, die zunächst angelegt wird. Dazu wird ein Rechtsklick auf dem Projekt gemacht und „New _> Other…“ gewählt.



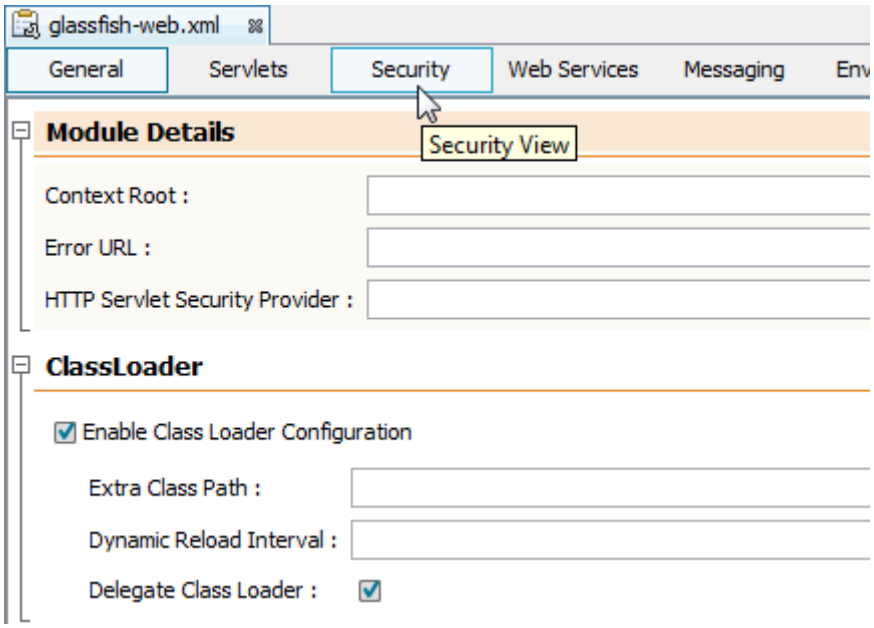
Links wird „GlassFish“ und rechts „GlassFish Descriptor“ gewählt und „Next>“ geklickt.



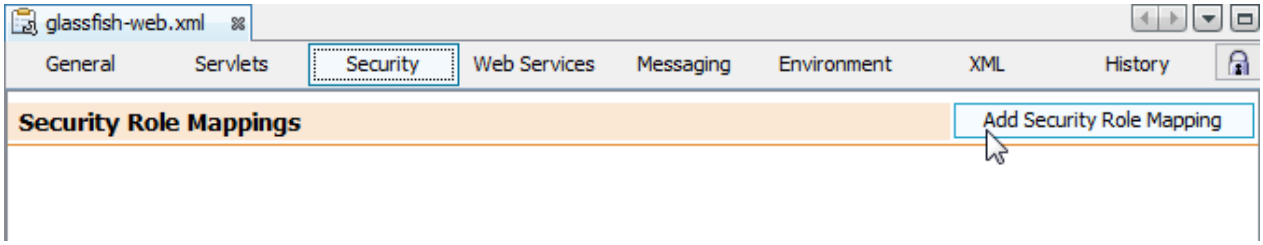
Es wird „Finish“ geklickt.



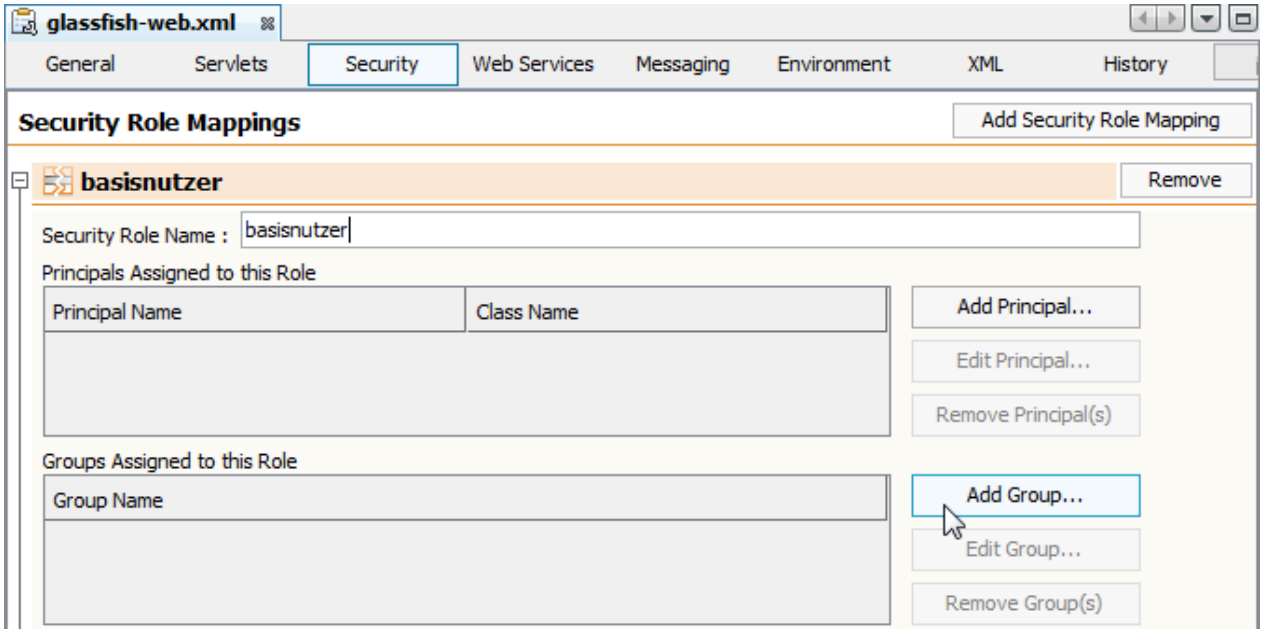
Es wird in der oberen Leiste auf „Security“ geklickt.



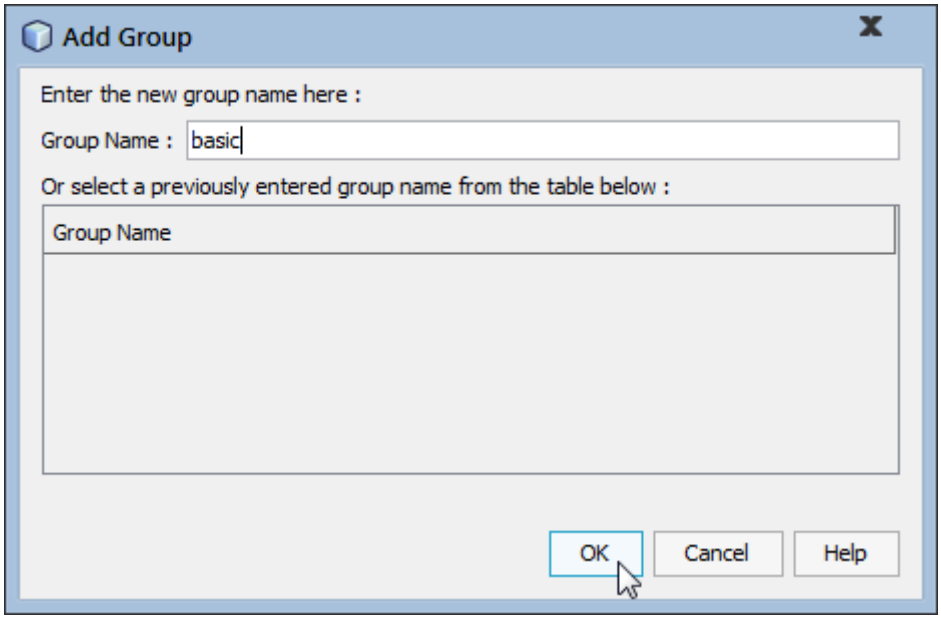
Es wird „Add Security Role Mapping“ geklickt.



Es wird ein „Security Role Name“ vergeben und auf „Add Group…“ geklickt.



Der Gruppenname „basic“ wird eingetragen, der auch in den SQL-Tabellen als Gruppenname verwendet wurde und mit „OK“ bestätigt. Man kann weitere „Security Role Mappings“ anlegen und auch mehrere Gruppen zuordnen.

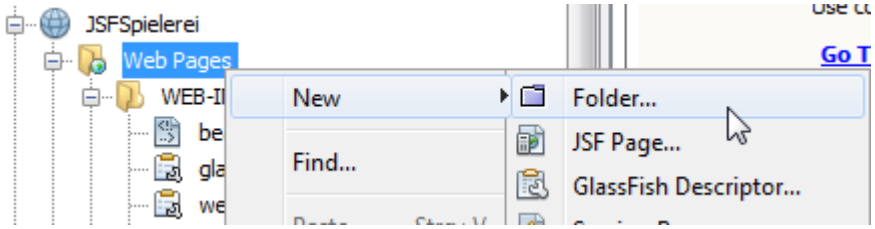


Die resultierende Datei kann wie folgt aussehen.

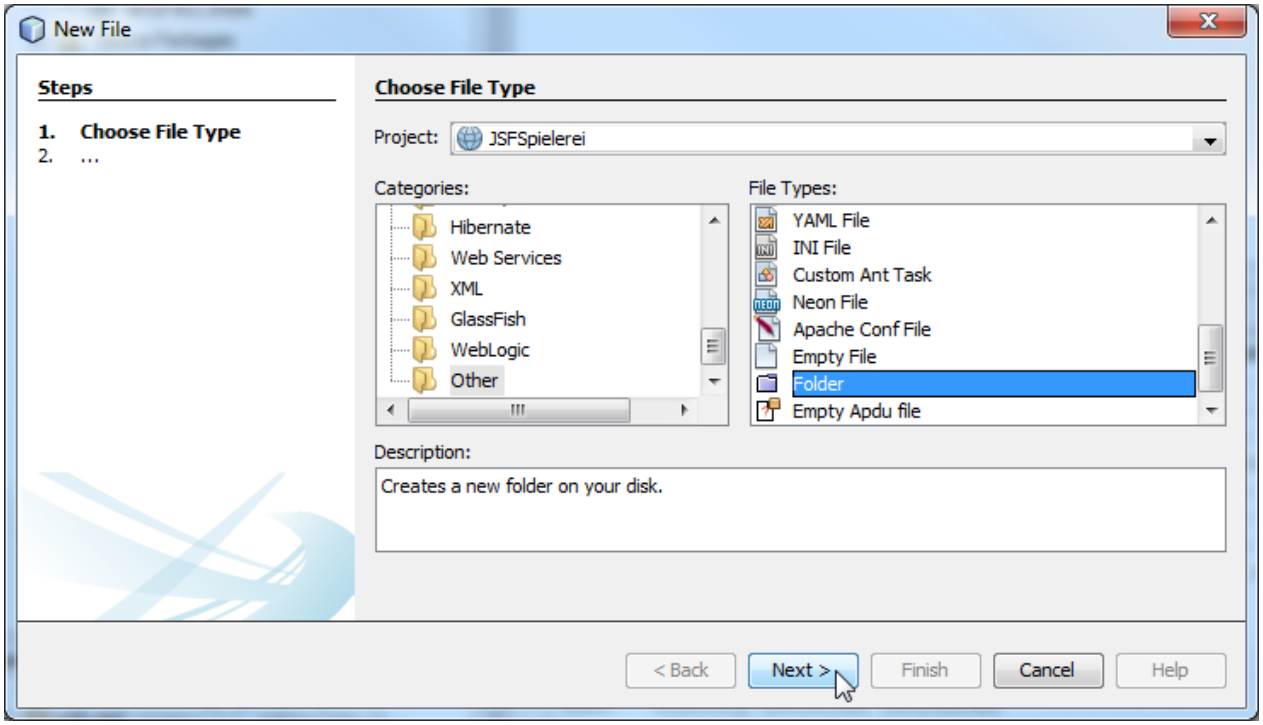
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1
Servlet 3.0//EN" "http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app error-url="">
  <security-role-mapping>
    <role-name>basisnutzer</role-name>
    <group-name>basic</group-name>
  </security-role-mapping>
  <class-loader delegate="true"/>
  <jsp-config>
    <property name="keepgenerated" value="true">
      <description>Keep a copy of the generated servlet class' java code.</description>
    </property>
  </jsp-config>
</glassfish-web-app>
```

10.7.6 Einstellungen in JSF

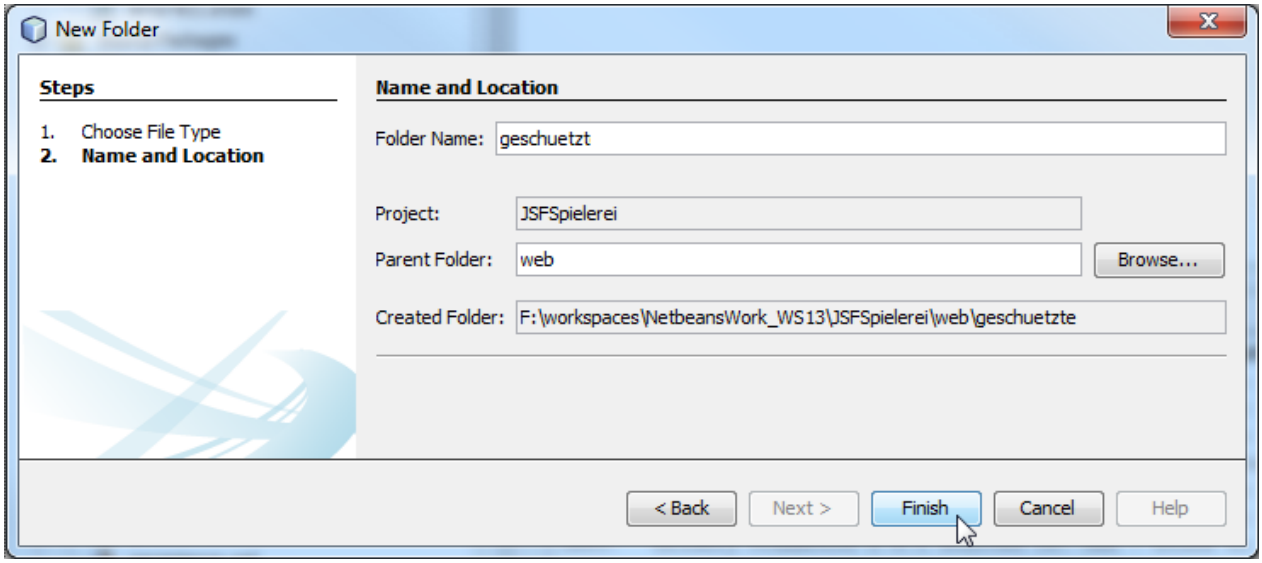
Generell kann man einfach alle Seiten schützen, was aber oft nicht sinnvoll ist, da man z. B. Nutzungshinweise oder Fehlermeldungen sichtbar machen möchte. Aus diesem Grund werden die Web-Seiten hier in einen Unterordner verschoben, dessen Inhalte geschützt werden sollen. Insgesamt kann so eine recht komplexe Zugriffsarchitektur aufgebaut werden, man sollte sich aber möglichst auf einzelne Seiten pro Nutzergruppe gbeschränken. Zunächst macht man einen Rechtsklick auf „Web Pages“ und wählt entweder direkt unter „New>“ dann „Folder…“ aus, oder muss über „Other…“ dahin steuern.



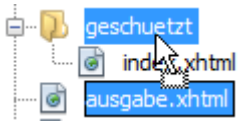
Über „Other…“ muss man auf der linken Seite wieder „Other“ und dann rechts „Folder“ wählen.



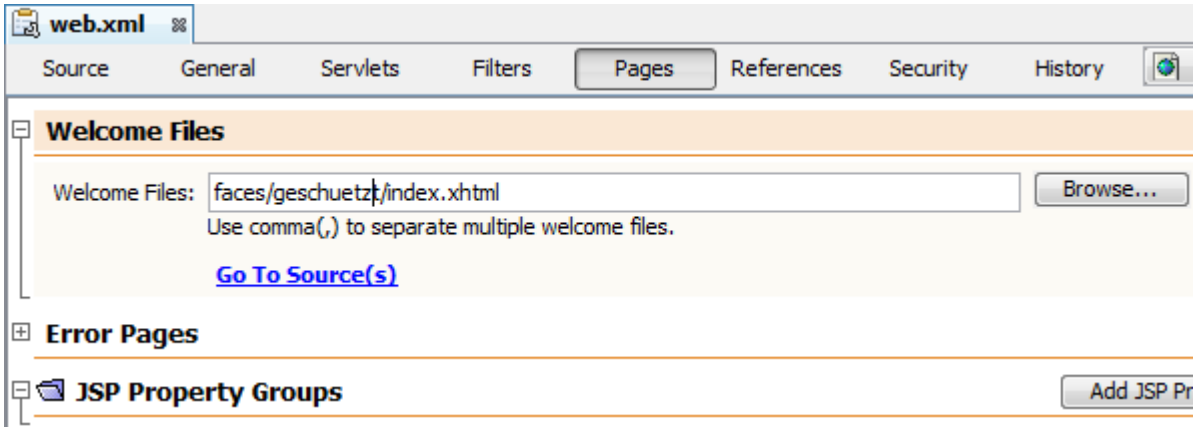
Man gibt dann einen beliebigen plausiblen Namen ein und drückt „Finish“.



Die einzelnen Seiten können einfach per Drag-and-Drop in den neuen Ordner geschoben werden.



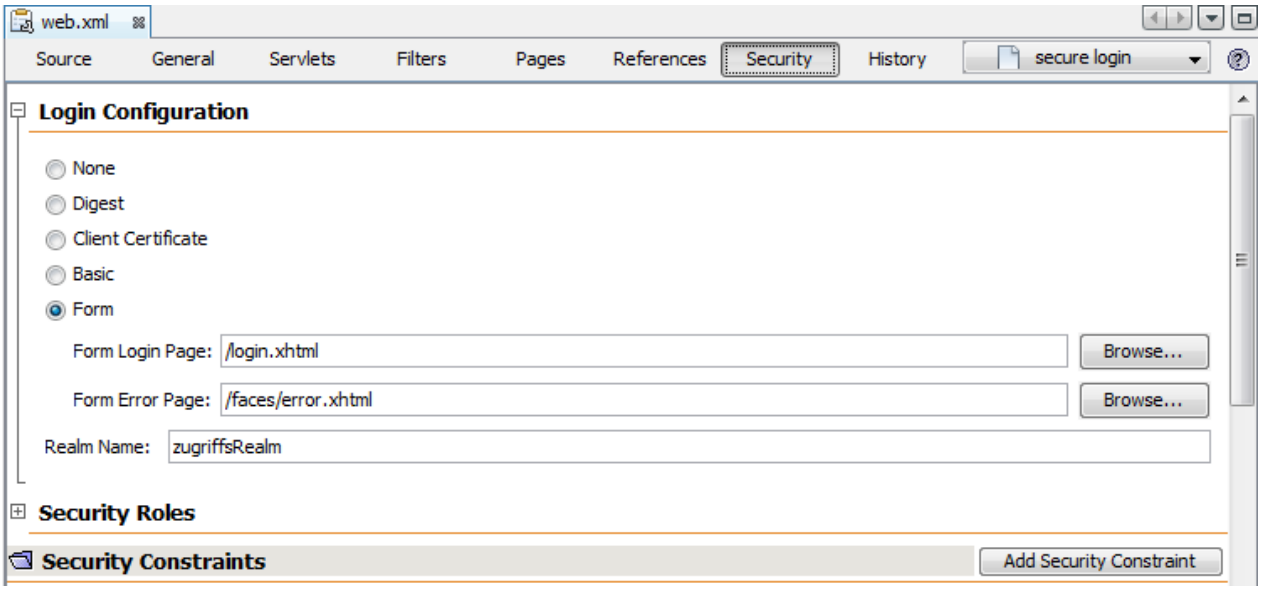
Nun muss auch in der web.xml die Startseite verändert werden. Man geht auf den Reiter „Pages“ und bearbeitet den Eintrag unter „Welcome Files“.



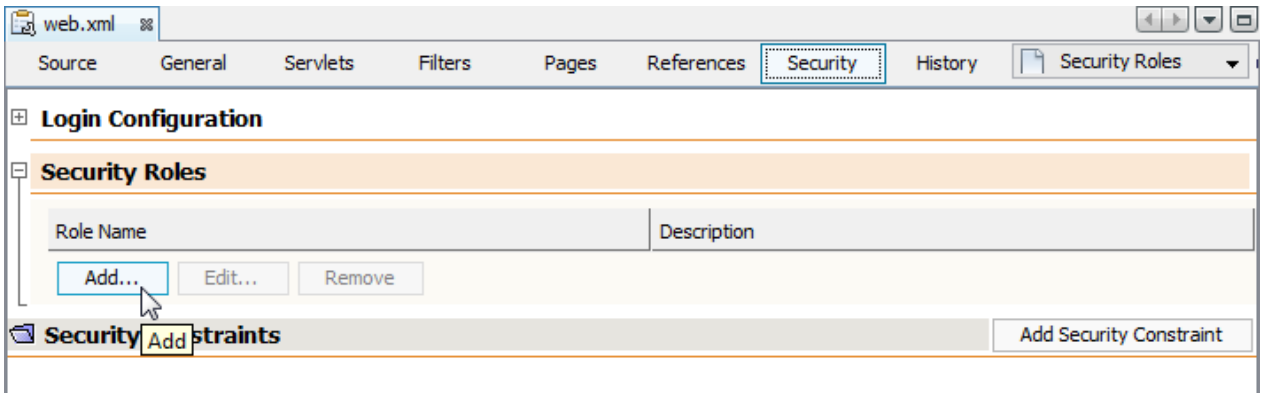
Nun muss angegeben werden, dass auf die Seiten im Unterordner geschuetzt nur eine bestimmte Nutzergruppe zugreifen kann.

Dazu wird in der web.xml „Security“ gewählt, „Login Configuration“ aufgeklappt, „Form“ ausgewählt, eine noch zu erstellende Seite zum Einloggen, eine noch zu erstellende Seite bei nicht erfolgreicher Anmeldung und der zuvor eingerichtete Realm eingetragen. Man beachte, dass bei diesem Ansatz die Login-Seite nicht geschützt ist, was kein Sicherheitsproblem ist, allerdings ohne

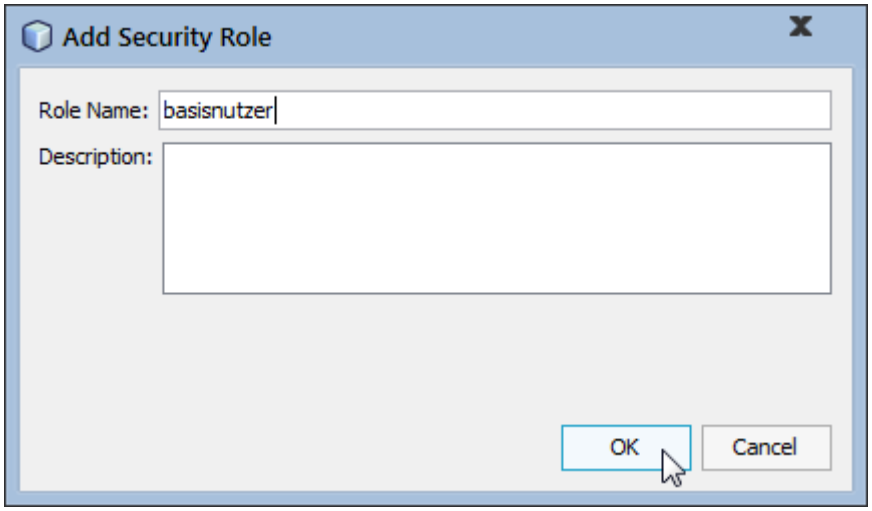
Beachtung zu technischen Problemen führen kann und später behandelt wird.



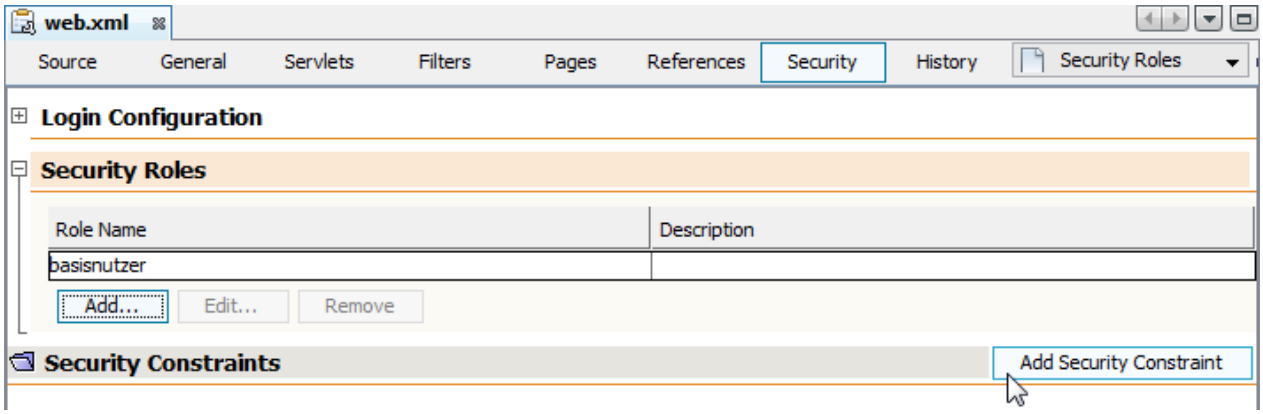
Dann wird unter „Security Roles“ auf „Add“ geklickt.



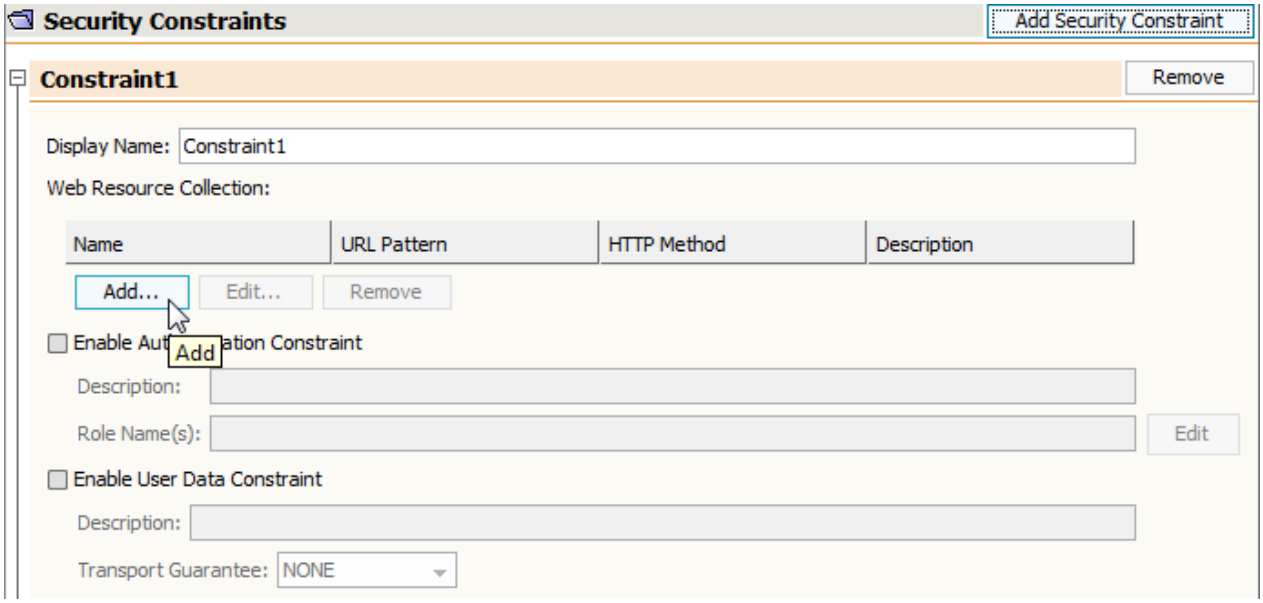
Die vorher erzeugte Security Role wird eingetragen.



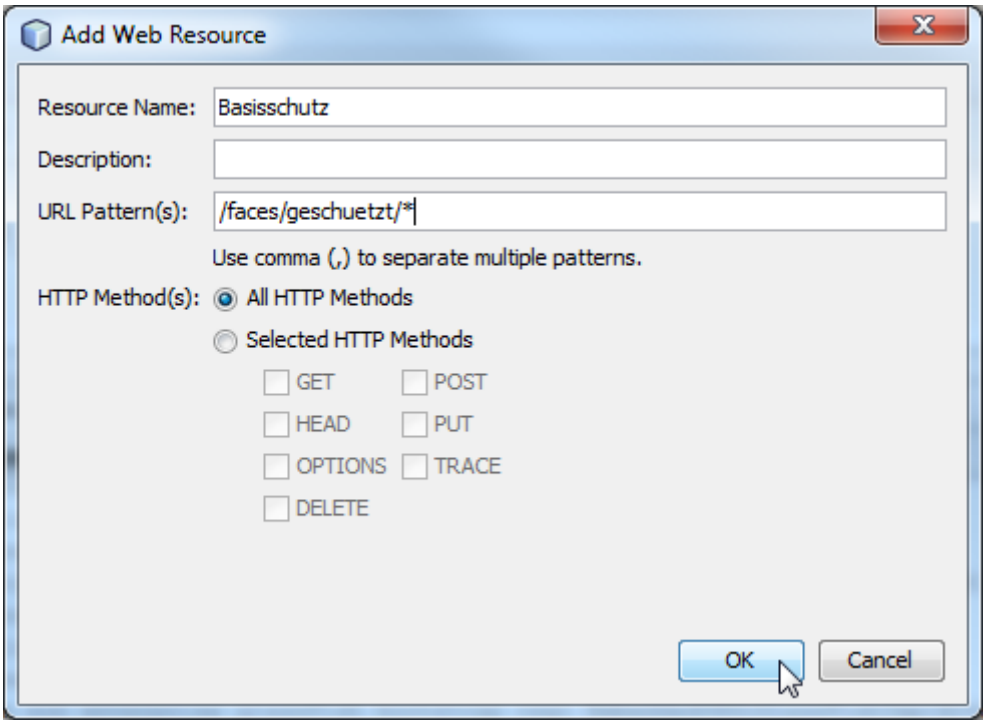
Danach wird unter „Security Constraints“ auf „Add Security Constraint“ geklickt.



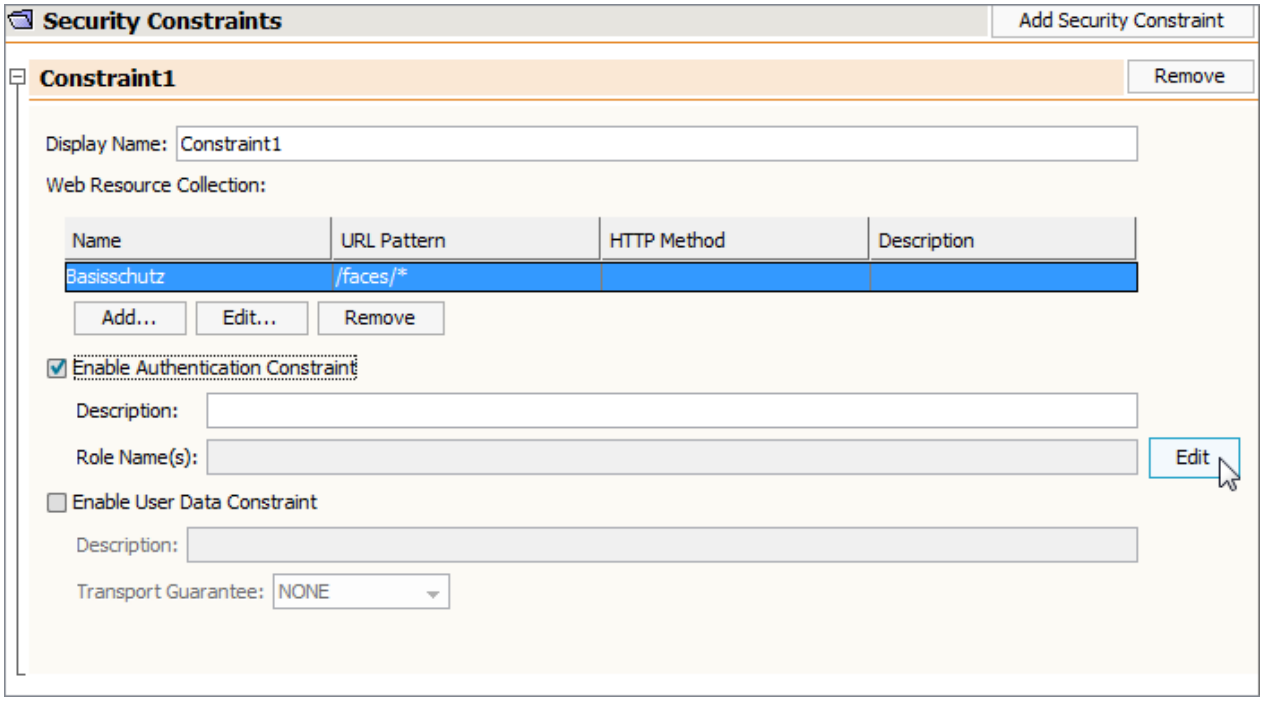
Es wird auf „Add…“ geklickt.



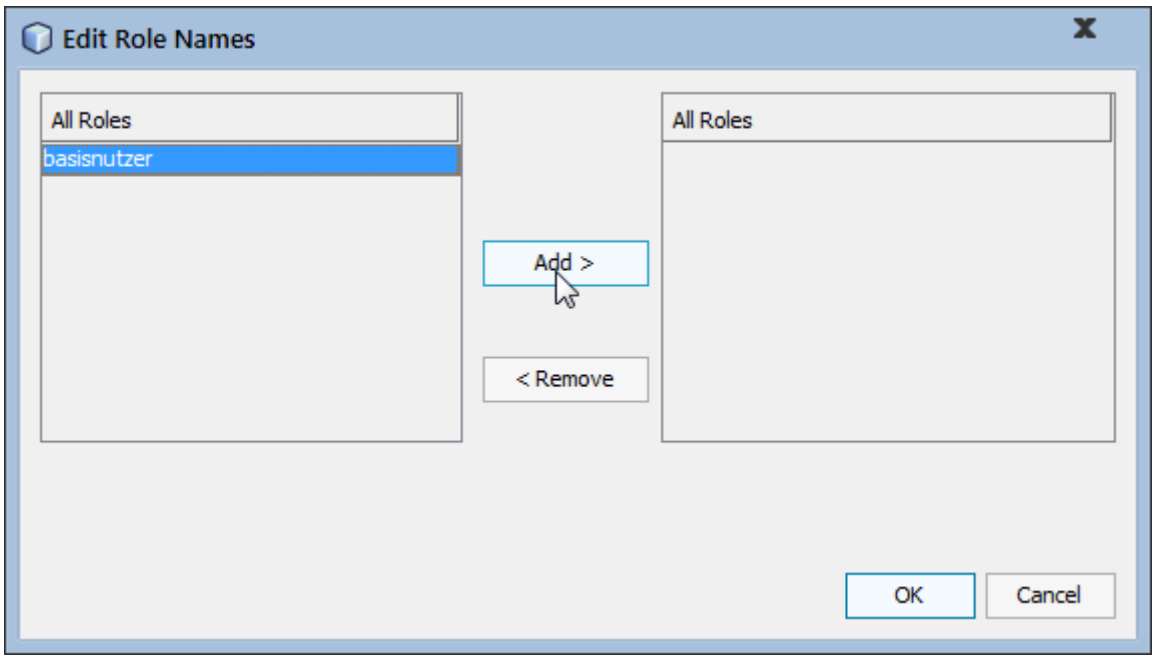
Im Beispiel sollen alle Seiten geschützt sein, was am bearbeiteten Pfadeintrag sichtbar wird.



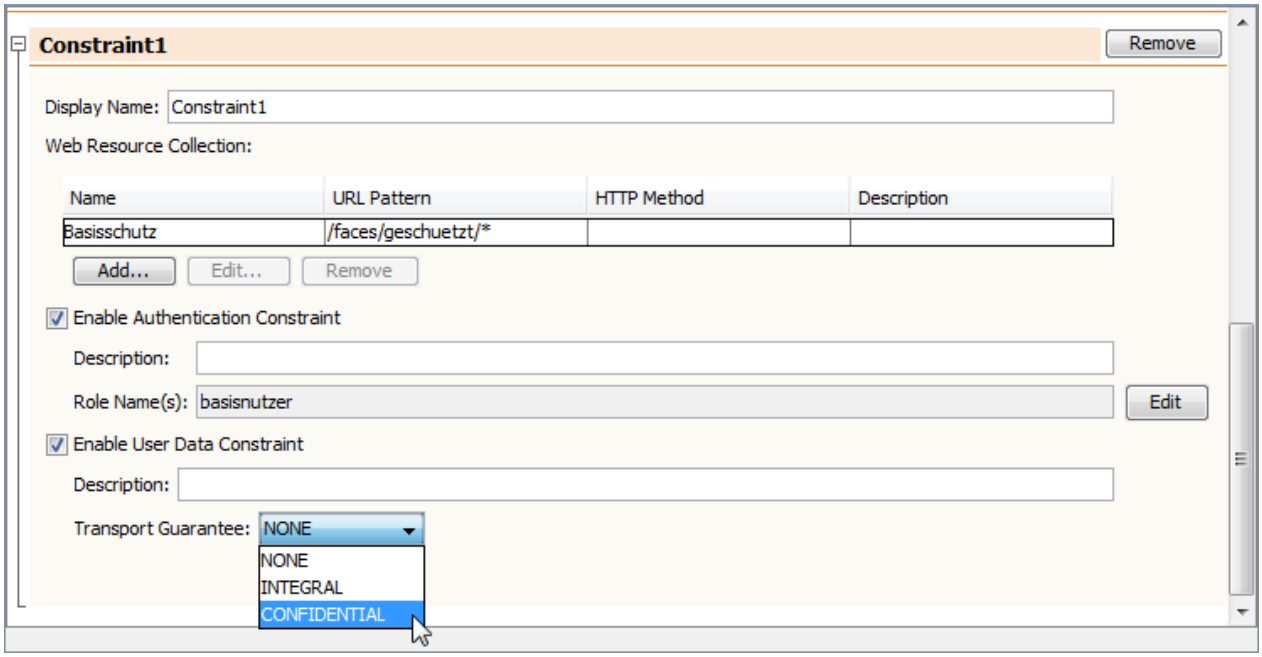
Danach wird das Constraint mit einem Haken links „Enable Authentification Constraint“ markiert und rechts auf „Edit“ geklickt.



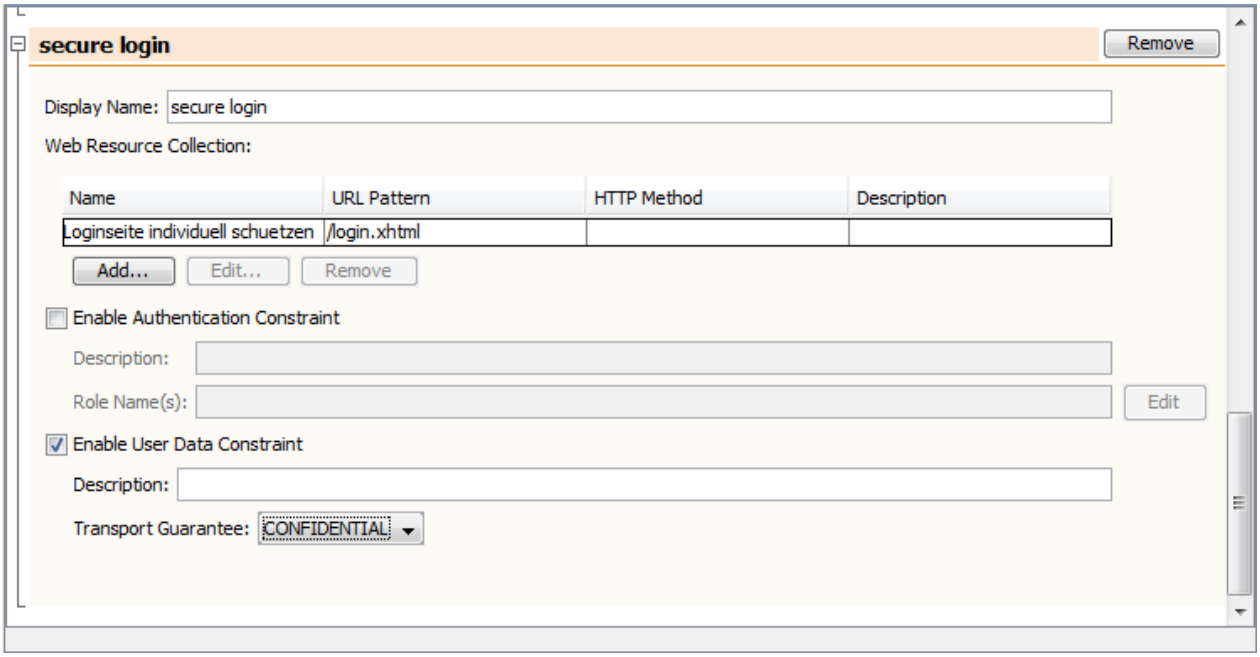
Die Rolle „basisnutzer“ wird links angeklickt, auf „Add>“ geklickt und die Aktion mit „OK“ abgeschlossen.



Dann wird der Haken bei „Enable User Data Constraint“ und die „Transport Guarantee“ auf „CONFIDENTAL“ gesetzt. Diese Garantie stimmt nicht ganz, da wir kein Sicherheitszeritikat haben, was eine kostenpflichtige Zertifizierung beinhalten würde.



Um die Login-Seite ordentlich einzubinden, wird hier eine zweite Security Constraint hinzugefügt. Deren Einträge der nachfolgenden Abbildung entnommen werden können. Man erkennt, dass auch einzelne Seiten geschützt werden können und man auf Rollennamen verzichten kann, wenn die Regel generell gilt.



Die resultierende Datei web.xml sieht dann wie folgt aus.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app version="3.1"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/geschuetzt/index.xhtml</welcome-file>
    </welcome-file-list>
    <security-constraint>
        <display-name>Constraint1</display-name>
        <web-resource-collection>
            <web-resource-name>Basisschutz</web-resource-name>
            <description/>
            <url-pattern>/faces/geschuetzt/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <description/>
            <role-name>basisnutzer</role-name>
        </auth-constraint>
        <user-data-constraint>
            <description/>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
    <security-constraint>
        <display-name>secure login</display-name>
        <web-resource-collection>
            <web-resource-name>Loginseite individuell schuetzen</web-resource-name>
            <description/>
            <url-pattern>/login.xhtml</url-pattern>
        </web-resource-collection>
        <user-data-constraint>
            <description/>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
```

```
        </user-data-constraint>
    </security-constraint>
    <login-config>
        <auth-method>FORM</auth-method>
        <realm-name>zugriffsRealm</realm-name>
        <form-login-config>
            <form-login-page>/login.xhtml</form-login-page>
            <form-error-page>/faces/error.xhtml</form-error-page>
        </form-login-config>
    </login-config>
    <security-role>
        <description/>
        <role-name>basisnutzer</role-name>
    </security-role>
</web-app>
```

Nun muss noch die Seite login.xhtml im Ordner „Web Pages“ angelegt werden, deren Einträge zwar vom Layout, aber nicht bei der aufgerufenen Aktion noch den Parameternamen verändert werden dürfen.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE     html     PUBLIC     "-//W3C//DTD     XHTML     1.0     Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Einloggen</title>
    </h:head>

    <h:body>
        <form name="POST" action="j_security_check">
            Nutzer:<input type="text" name="j_username"
                          value="" size="15" /><br/>
            Passwort:<input type="text" name="j_password"
                            value="" size="15" /><br/>
            <input type="submit" value="Anmelden" />
        </form>
    </h:body>
</html>
```

Abschließend wird noch die Fehlerseite error.xhtml im Ordner „Web Pages“ angelegt.

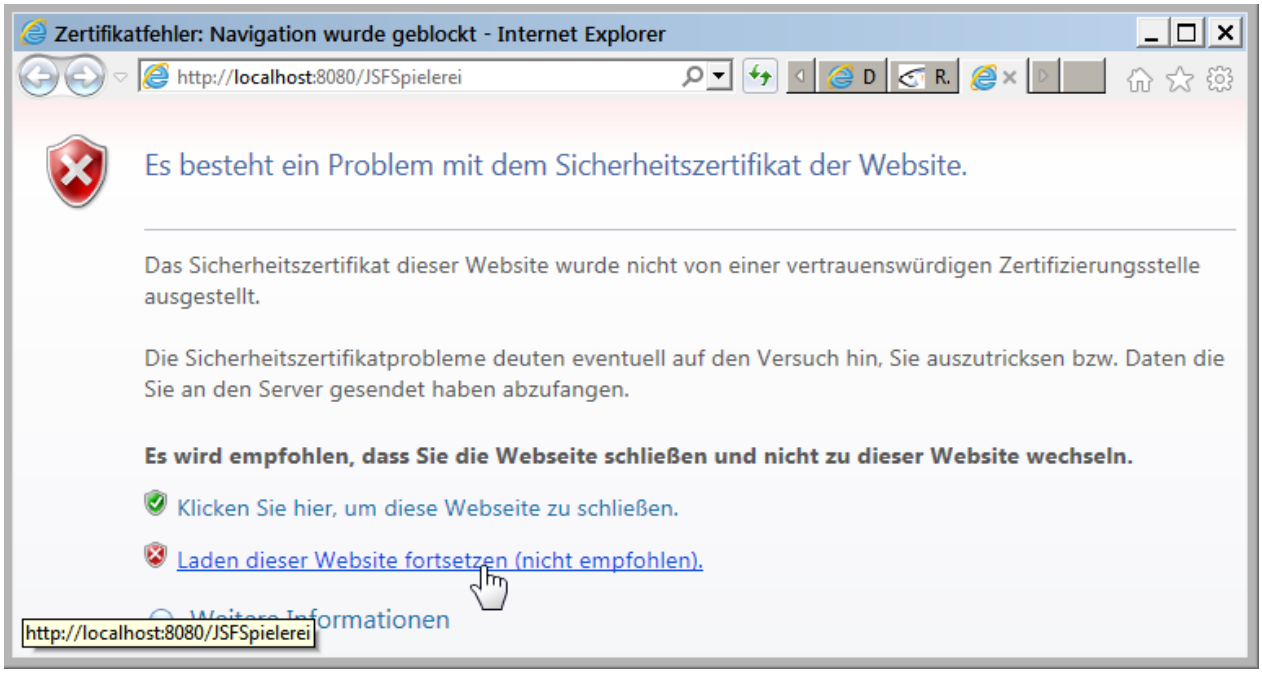
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE     html     PUBLIC     "-//W3C//DTD     XHTML     1.0     Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Fehlende Berechtigung</title>
    </h:head>
    <h:body>
```

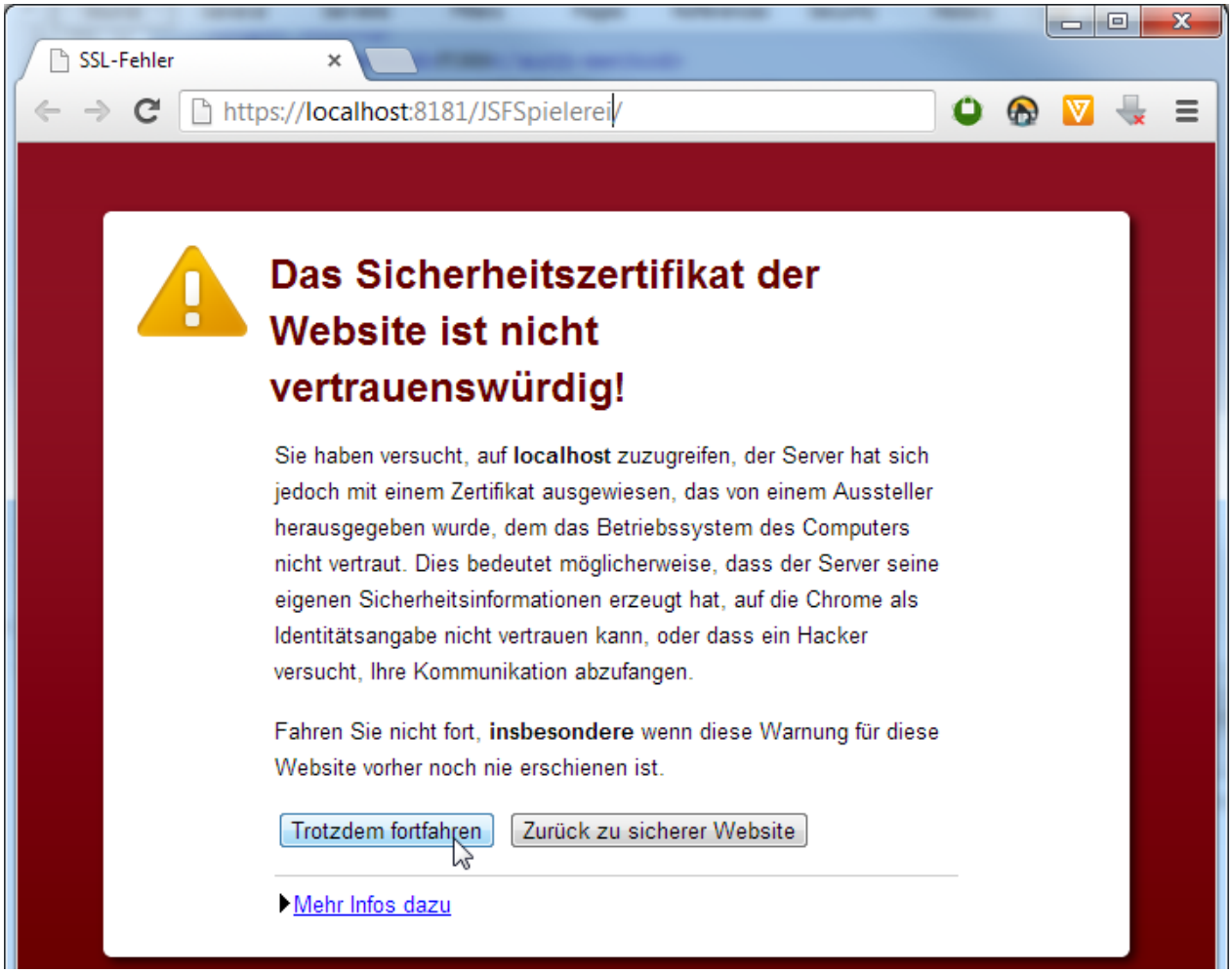
```
            <h:form>
                <h:outputText
                    value="Hey, Du kommst nit rein "/>
                <h:commandButton value="Zum Start"
                                 action="#{modul.sessionBeenden}"/>
            </h:form>
        </h:body>
    </html>
```

Zusätzlich wird in der Klasse Modul.java noch folgende Methode ergänzt.

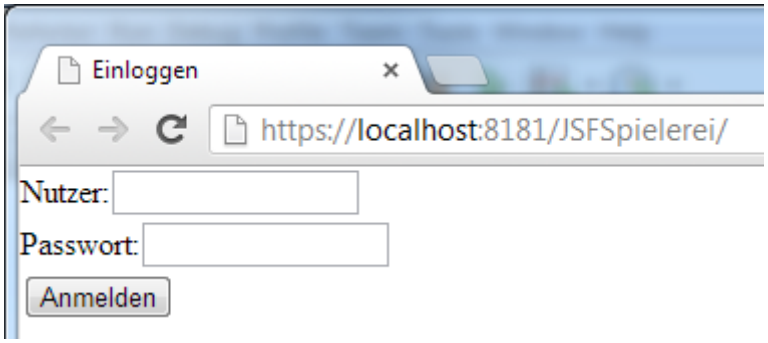
```
public String sessionBeenden() {
    ExternalContext ectx = FacesContext
            .getCurrentInstance().getExternalContext();
    ectx.invalidateSession();
    return "/geschuetzt/index.xhtml?faces-redirect=true";
}
```

Startet man dann die Applikation, wird man zunächst auf das fehlende Zertifikat hingewiesen. Der Hinweis wird über „Trotzdem fortfahren“ ignoriert.



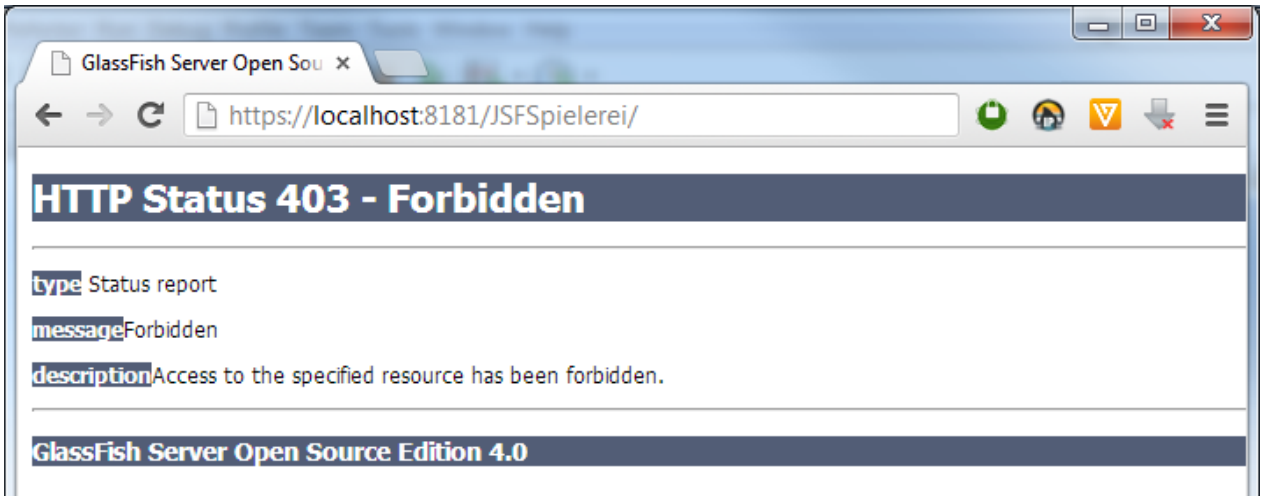


sofort zur Authentifizierung aufgefordert. Nur edna und uwe können sich einloggen.

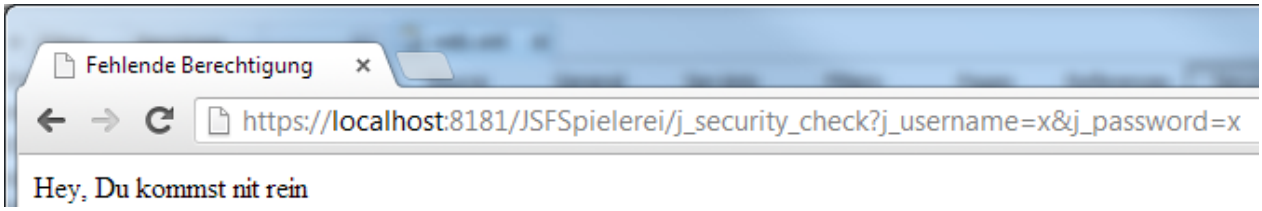


Die Authentifizierung sieht abhängig vom Browser etwas unterschiedlich aus.

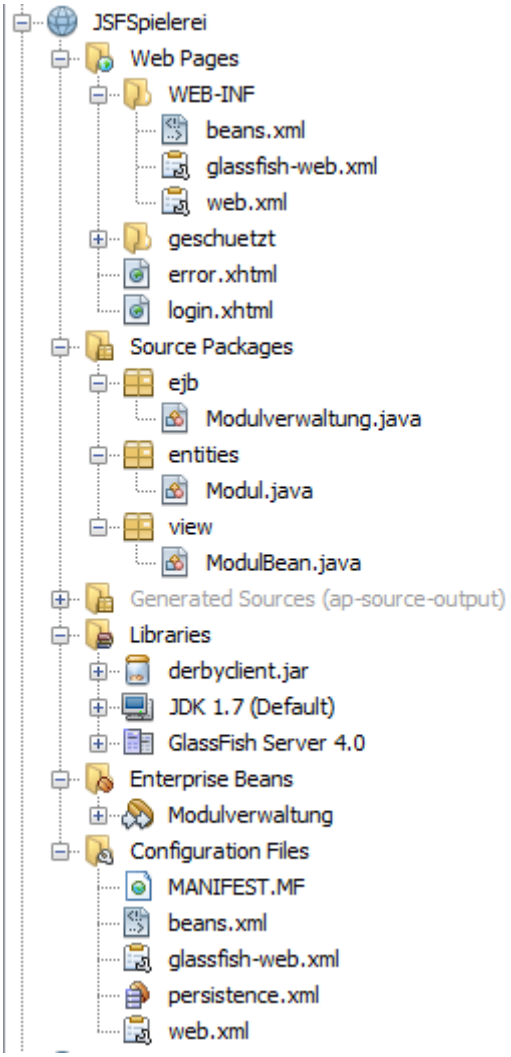
Meldet sich otto an, der zwar Zugang zum Server, aber nicht zu dieser Applikation hat, wird ein HTTP 403-Fehler ausgegeben, der in JSF durch eine eigene Seite ersetzt werden muss, was man in der web.xml einstellen kann (im nächsten Unterkapitel behandelt). Noch schlimmer, kann man ohne weitere Schritte den Nutzer nicht abmelden, es fehlt die Möglichkeit, die Session zu beenden. Aus diesem Grund ist es sinnvoll, die Sessiondauer (<session-timeout>) in der web.xml von 30 Minuten z. B. auf eine Minute herunter zu setzen.



Meldet sich ein anderer Nutzer falsch an, kommt er zur Fehlerseite. Man erkennt, dass es sinnvoll wäre, einen Link zurück zur Applikation zu ergänzen.



Das Projekt hat insgesamt dann folgenden Aufbau.



10.8 Behandlung mit Fehlerseiten

Im vorherigen Abschnitt konnte ein für den Server registrierter Nutzer nicht auf die Applikation zugreifen und bekam die Standard-403-Fehlerseite des Servers. Noch schlimmer kann man mit dem Browser durch den erneuten Aufruf der Startseite nicht erneut zur Einlog-Seite kommen, da der Browser sich den Nutzer in einer Session gemerkt hat. Ein Zugriff ist erst wieder nach Ablauf der Session, was in der web.xml konfigurierbar ist, möglich, wenn die Session nicht explizit beendet wird, was auch beim Ausloggen sehr sinnvoll ist. In diesem Abschnitt wird dazu beispielhaft der 403-Fehler behandelt und eine Möglichkeit zur Session-Invalidierung gezeigt.

Zunächst wird eine Fehler-Seite mit beliebigen Namen angelegt. Hier wird die Seite error.xhtml dazu ergänzt, die den folgenden Inhalt dann hat. Durch Drücken des Knopfes soll auch die Session invalidiert werden.

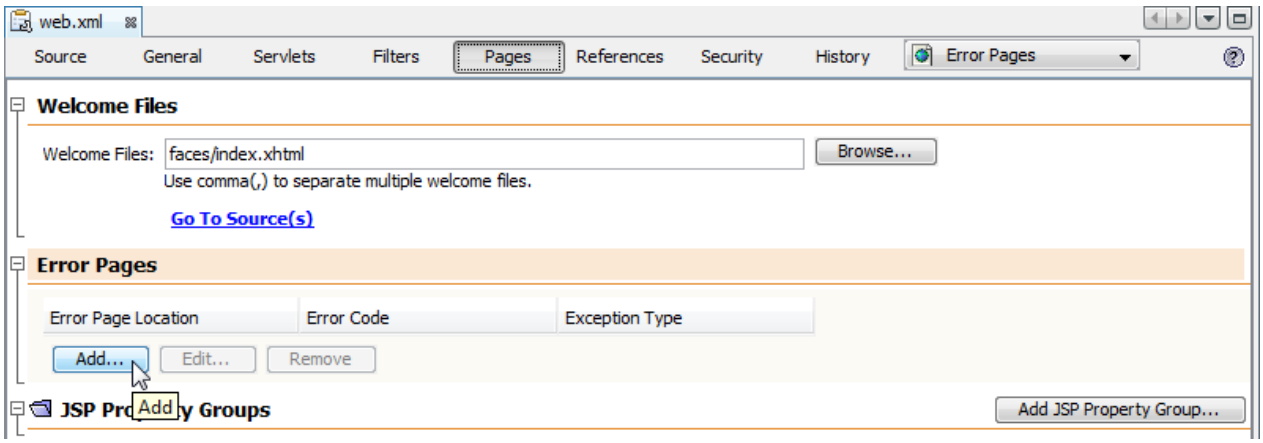
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Fehlende Berechtigung</title>
    </h:head>
    <h:body>
        <h:form>
            <h:outputText
                value="Hey, Du kommst nit rein "/>
            <h:commandButton value="Zum Start"
                             action="#{modul.sessionBeenden}"/>
        </h:form>
    </h:body>
</html>
```

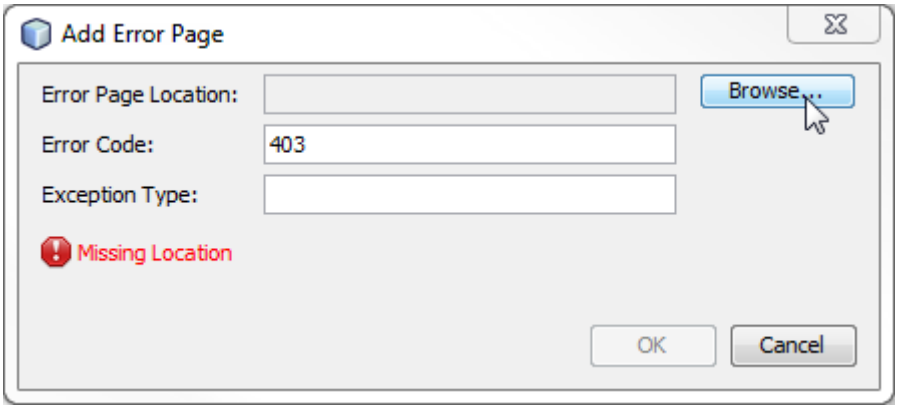
In der Klasse ModulBean muss die dazu passende Methode ergänzt werden. Man sieht, dass man auch in JSF auf das dahinter liegende Servlet zugreifen kann, was einem weiteren Möglichkeiten, wie den Zugriff auf Parameter ermöglicht. Generell sollte man aber sehr beacht mit diesem Griff in die „Innereien“ umgehen. Man beachte auch das hier sogar notwendige ?faces-redirect=true mit dem sonst nur garantiert wird, dass auch der richtige Link-Name im Browser angezeigt wird.

```
public String sessionBeenden() {
    ExternalContext ectx = FacesContext
            .getCurrentInstance().getExternalContext();
    ectx.invalidateSession();
    return "/geschuetzt/index.xhtml?faces-redirect=true";
}
```

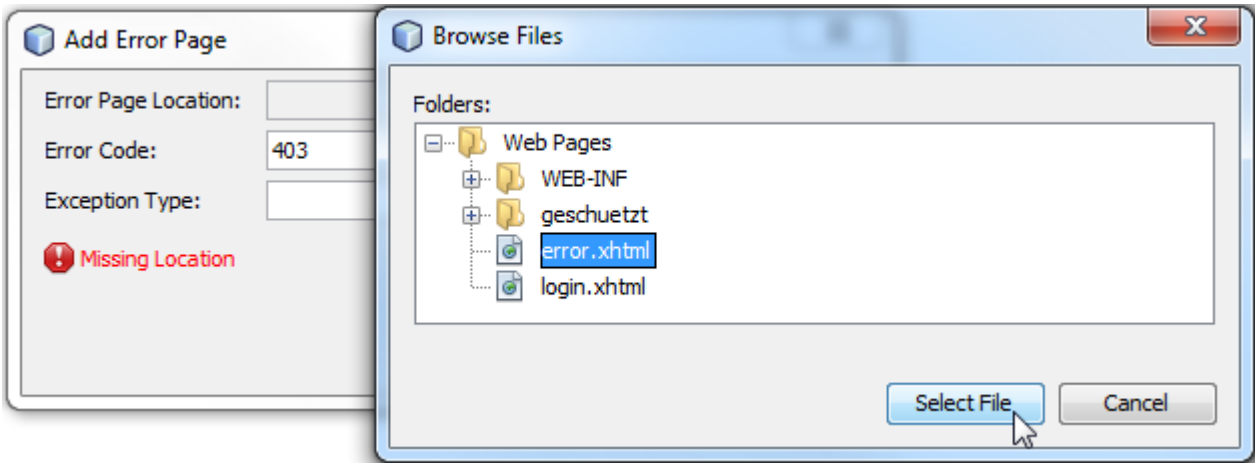
Die Fehlerseite muss jetzt in der web.xml bekannt gemacht werden. Dazu geht auf den Reiter „Pages“ und drückt im Bereich „Error Pages“ auf „Add…“.



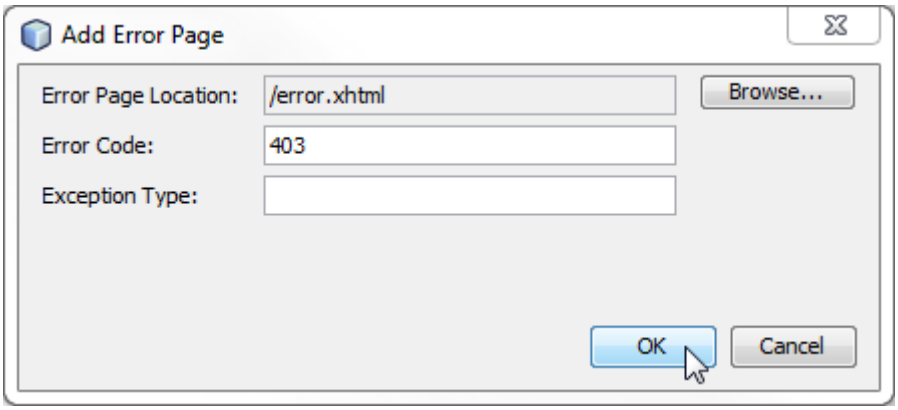
Man trägt die Fehlernummer (Error Code) ein und klickt rechts-oben auf „Browse…“.



Man steuert hier zur angelegten Fehlerseite und drückt „Select File“.



Man beendet die Ergänzung mit „OK“.



In der web.xml wurden folgende Zeilen nicht ganz korrekte Zeilen ergänzt.

```
<error-page>
```

```
    <error-code>403</error-code>
    <location>/faces/error.xhtml</location>
</error-page>
```

Man muss In der web.xml einen Teilpfad ändern und /faces davorsetzen.

```
<error-page>
    <error-code>403</error-code>
    <location>/faces/error.xhtml</location>
</error-page>
```

Zum Hintergrund: NetBeans setzt als Standard fest, dass alle Dateuen im Unterbereich faces mit dem Faces-Servlet behandelt werden sollen, was wie folgt in der web.xml steht.

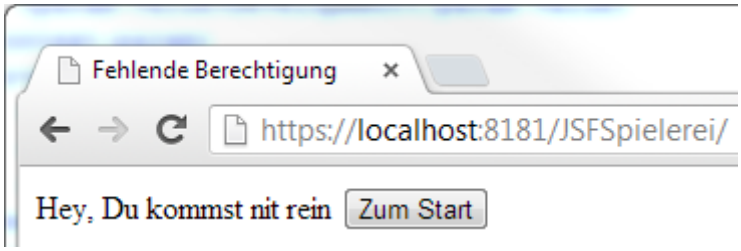
```
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
```

Aus diesem Grund steht immer das /faces vor den Seiten, wurde aber in NetBeans nicht bei den Fehlerseiten berücksichtigt, da hier auch einfach HTML-Seite genutzt werden können. Ein alternativer Ansatz ist es, das Faces-Servlet einfach für alle Dateien zu nutzen, die mit „.xhtml“ enden, was wie folgt aussieht.

```
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.xhtml</url-pattern>
</servlet-mapping>
```

Damit würde obiges Problem vermieden. Ob man diesen Ansatz nutzen möchte, muss man selbst bei der Projekteinrichtung entscheiden, da nachträgliche Änderungen sehr fehlerträchtig sind. Wird z. B. nur der Seitenname und kein oder kein sinnvoller Inhalt einer XHTML-Seite dargestellt, kann man davon ausgehen, dass die Seite nicht zum Faces-Servlet umgeleitet wird und es Pfadprobleme gibt.

Der Nutzer „otto“ kann jetzt problemlos wieder zur Startseite gelangen.



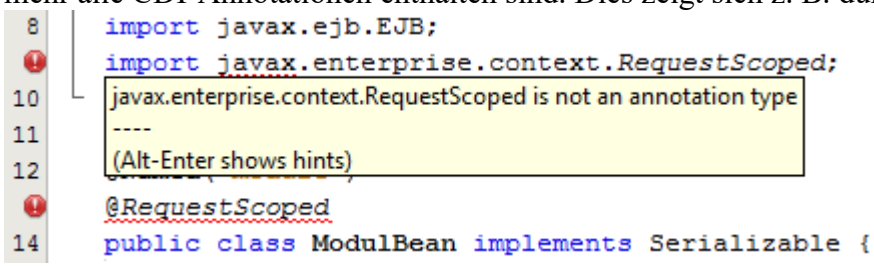
Die gleiche Methode kann natürlich auch zum Ausloggen aufgerufen werden, z. B. in index.xhtml.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
```

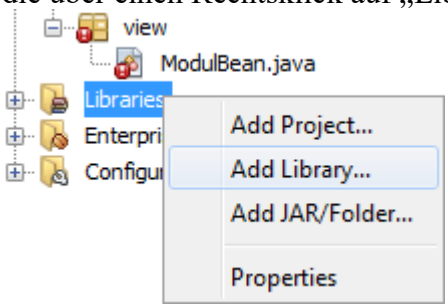
```
        <h:head>
            <title>Moduleingabe</title>
        </h:head>
        <h:body>
            <h:form>
                <h:outputText value="Modulname "/>
                <h:inputText id="mname" value="#{modul.name}"
                             required="true"/><br/>
                <h:outputText value="Modulnummer "/>
                <h:inputText id="mnr" value="#{modul.nr}"
                             required="true"/><br/>
                <h:commandButton value="Abschicken"
                                 action="#{modul.uebernehmen}"/>
                <h:commandButton value="Abmelden"
                                 action="#{modul.sessionBeenden}"/>
            </h:form>
        </h:body>
    </html>
```

10.9 Anmerkungen zu CDI in NetBeans 8.0.1

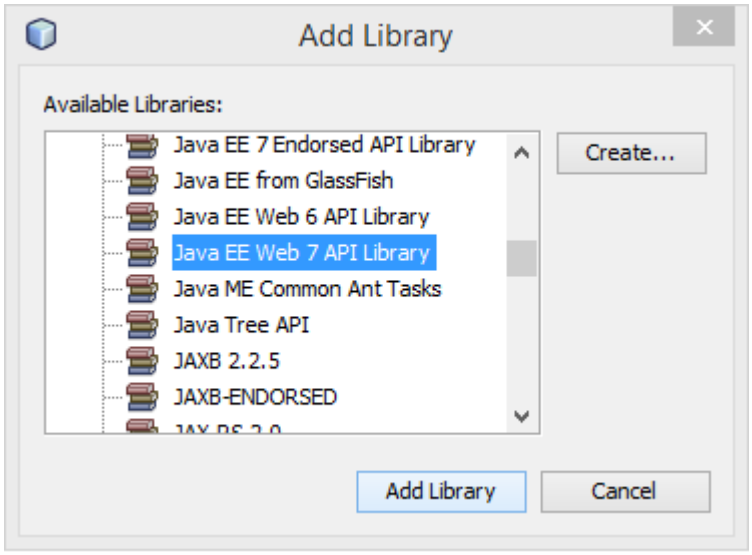
Bei der Nutzung von CDI fällt auf, dass in der GlassFish Server 4.1-Bibliothek anscheinend nicht mehr alle CDI-Annotationen enthalten sind. Dies zeigt sich z. B. durch folgende Fehlermeldung.



Eine Abänderung ist theoretisch dadurch möglich, dass man die gesamte CDI-Implementierung in den Libraries als jar-Dateien ergänzt. Zum Glück gibt es im GlassFish dazu schon eine Bibliothek, die über einen Rechtsklick auf „Libraries“ und der Auswahl „Add Library…“ auswählbar wird.



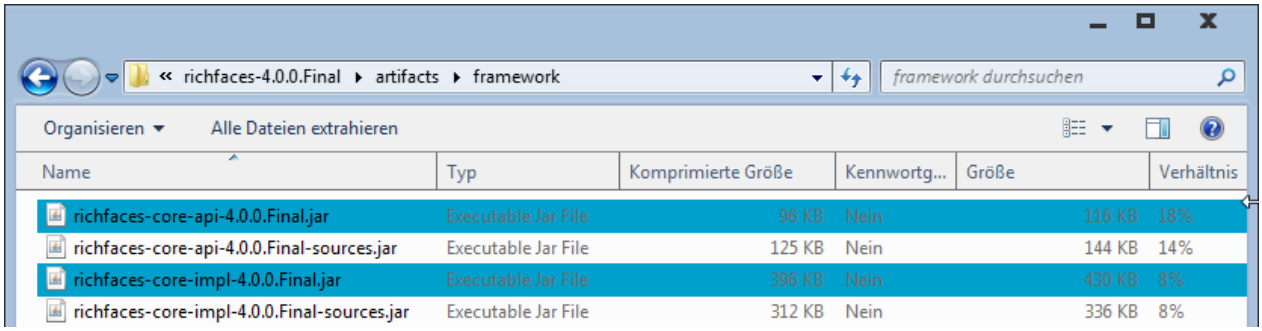
Es wird „Java EE Web 7 API Library“ mit „Add Library“ hinzugefügt.



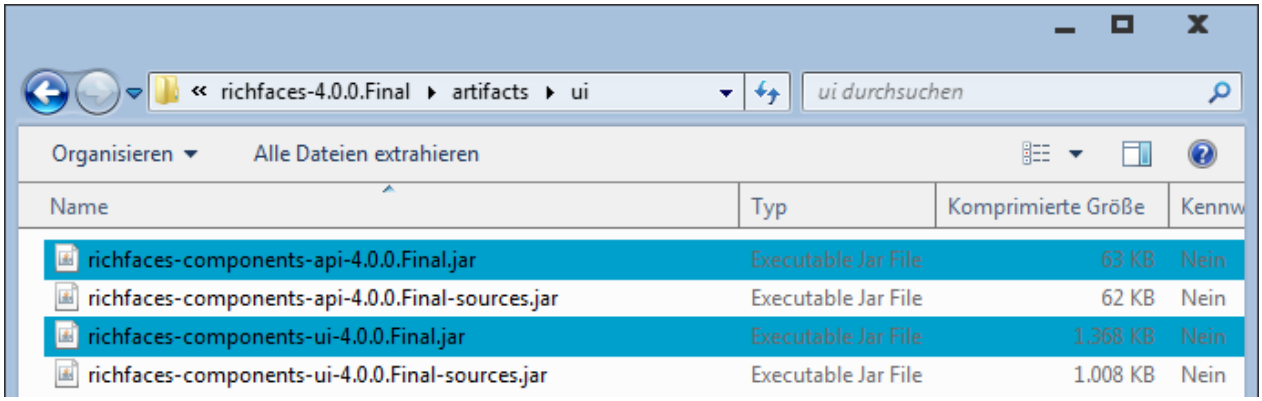
10.10 Nutzung von RichFaces 4.0.0 für JSF 2.0 (NetBeans 6.9)

Zunächst benötigt man die Implementierung von Richfaces, die man über die Seite http://www.jboss.org/richfaces/download.html erhält. Dabei sollte ein „Stable"-Download genutzt werden, z. B. http://downloads.jboss.org/richfaces/releases/4.0.X/4.0.0.Final/richfaces-4.0.0.Final.zip.

Innerhalb des Zip-Files existiert ein artifacts/framework-Verzeichnis. Die zwei enthaltenen jar-Dateien ohne Endung „-sources.jar" werden in ein Verzeichnis auf der Festplatte kopiert, z. B. ein lib-Verzeichnis auf der gleichen Höhe, wie die Projektverzeichnisse der NetBeans-Projekte.



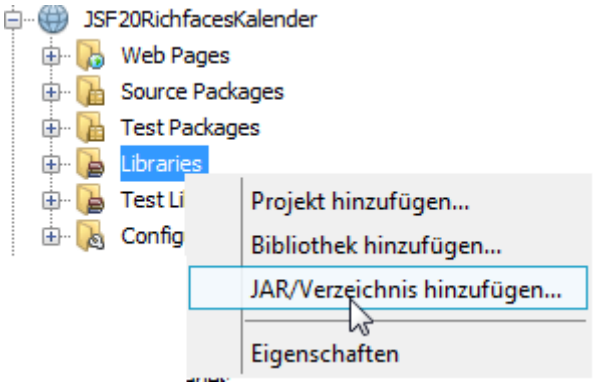
Dazu werden die zwei weiteren Dateien aus dem artifaces/ui-Verzeichnis benötigt, die auch nicht auf „-sources.jar" enden und in das gleiche lib-Verzeichnis kopiert.



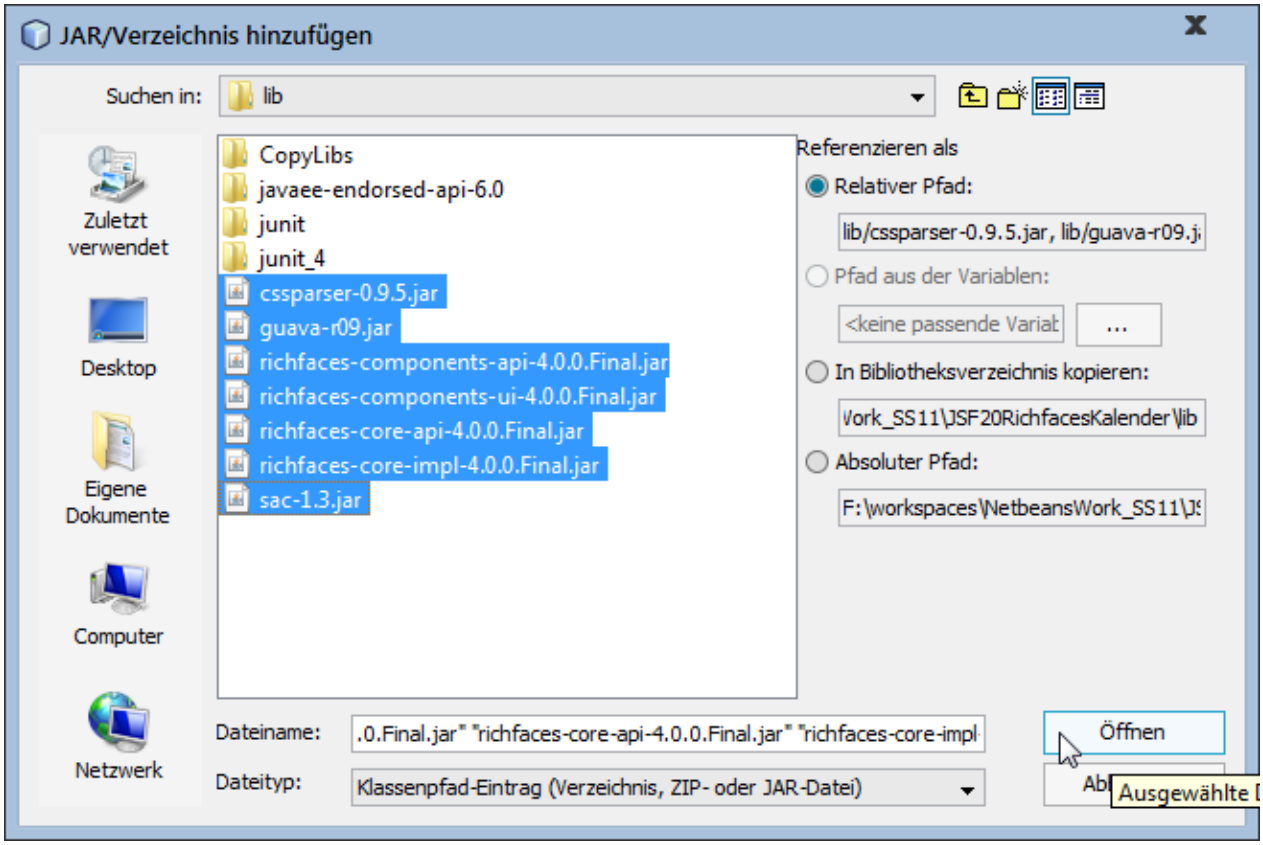
Weiterhin werden einige weitere Bibliotheken als jar-Dateien benötigt. Man muss beachten, dass die Versionsnummern zu den geforderten Versionsnummern passen, dabei können neue Versionen leider „zu neu“ und damit nicht kompatibel mit den alten Versionen sein. Für die hier verwandte RichFaces-Version werden folgende Bibliotheken benötigt. Diese jar-Dateien werden in das gleiche Verzeichnis, wie die jar-Dateien von Richfaces gelegt. Benötigt werden:

- cssparser.jar von der Seite http://repo1.maven.org/maven2/net/sourceforge/cssparser/cssparser/0.9.5/, oder direkt http://repo1.maven.org/maven2/net/sourceforge/cssparser/cssparser/0.9.5/cssparser-0.9.5.jar
- sac.jar, von der Seite http://repo1.maven.org/maven2/org/w3c/css/sac/1.3/, oder direkt http://repo1.maven.org/maven2/org/w3c/css/sac/1.3/sac-1.3.jar
- guava.jar, von der Seite http://code.google.com/p/guava-libraries/downloads/list, oder direkt http://guava-libraries.googlecode.com/files/guava-r09.zip

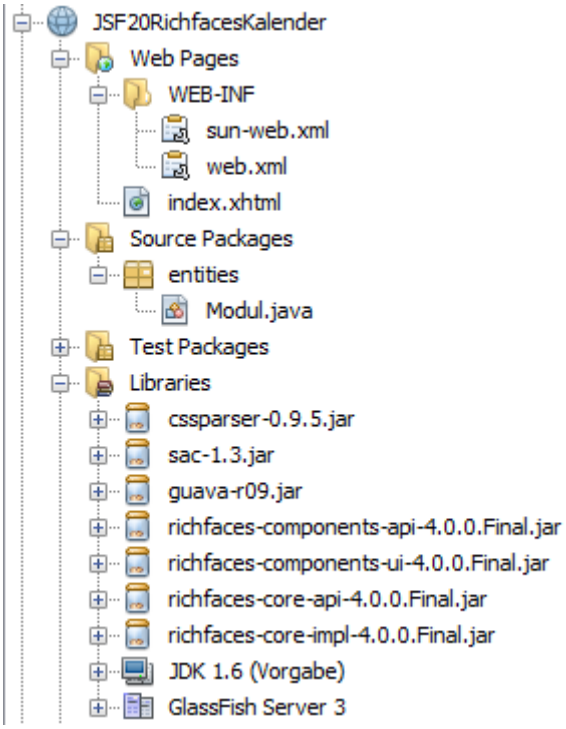
Es wird dann zunächst ein JSF-Projekt, wie am Anfang des Kapitels beschrieben, angelegt. Danach werden die jar-Dateien ergänzt. Dies geschieht durch einen Rechtsklick auf den Projektordner „Libraries“ in dem dann „JAR/Verzeichnis hinzufügen...“ ausgewählt wird.



Jetzt wird zum Verzeichnis mit den jar-Dateien manövriert und die unten angegebenen Dateien markiert. Arbeitet man z. B. auf einem USB-Stick empfiehlt sich rechts oben „Relative Path“ auszuwählen.



Danach kann man Richfaces direkt nutzen, es müssen keine Einstellungen in der web.xml vorgenommen werden. Es ist allerdings möglich, hier recht viele Parameter zu verändern, so dass bei einer intensiveren Einarbeitung auch die Änderung der web.xml eine wichtige Rolle spielt. Eine typische Projektstruktur kann wie folgt aussehen.



Im konkreten Beispiel wird der Kalender von Richfaces mit sehr wenigen seiner Einstellungsmöglichkeiten genutzt. Dazu wird die folgende Managed Bean angelegt, die ein Datum verwalten kann.

```
package entities;

import java.io.Serializable;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

@ManagedBean(name="modul")
@RequestScoped
public class Modul implements Serializable {

  private Date start=new Date();

  public Modul(){}

  public Date getStart() {
    System.out.println("getstart "+start+" ..");
    return start;
  }

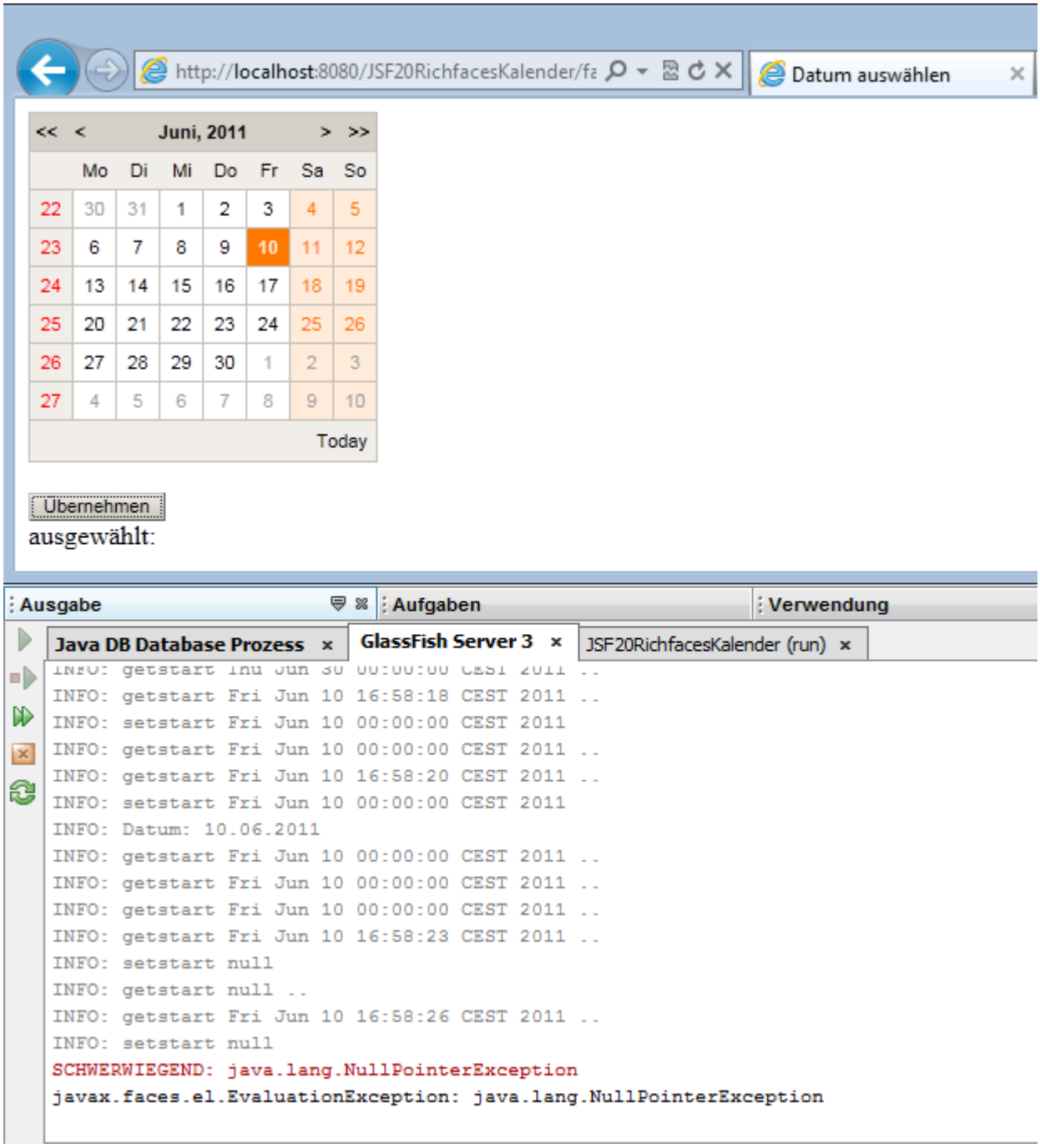
  public void setStart(Date start) {
    System.out.println("setstart "+start);
    this.start = start;
  }

  public String show(){
    System.out.println("Datum: "+new SimpleDateFormat("dd.MM.yyyy").format(start));
```

```
        return "";  // unsauber, bleibt auf Seite
      }
    }
```

In der folgenden Web-Seite wird deutlich, wie auf die Richfaces-Komponenten durch die Erweiterung des Namespaces zugegriffen werden kann. Beim Experimentieren sollte man sich nebenbei überlegen, wie man das Programm mit einer NullPointerexception abstürzen lassen kann ☺ (Programmierfehler, kein Richfaces-Fehler).

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
    <h:head>
        <title>Datum auswahlen</title>
    </h:head>
    <h:body>
        <h:form>
            <rich:calendar id="cal" value="#{modul.start}" popup="false"
                           datePattern="dd.MM.yyyy">
                <a4j:ajax  event="change" render="o"/>
            </rich:calendar>
            <br/>
            <h:commandButton action="#{modul.show}" value="Ubernehmen"/> <br/>
            <h:outputLabel value="ausgewahlt: "/>
            <h:outputText id="o" value="#{modul.start}" /><br/>
        </h:form>
    </h:body>
</html>
```

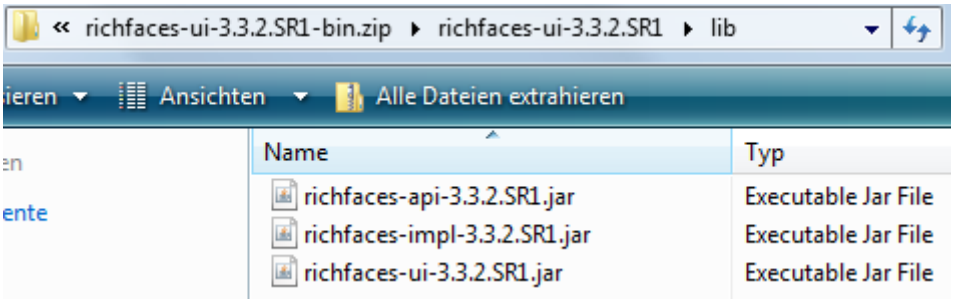


10.11 Nutzung von RichFaces 3.3.2 für JSF 1.2 (NetBeans 6.7)

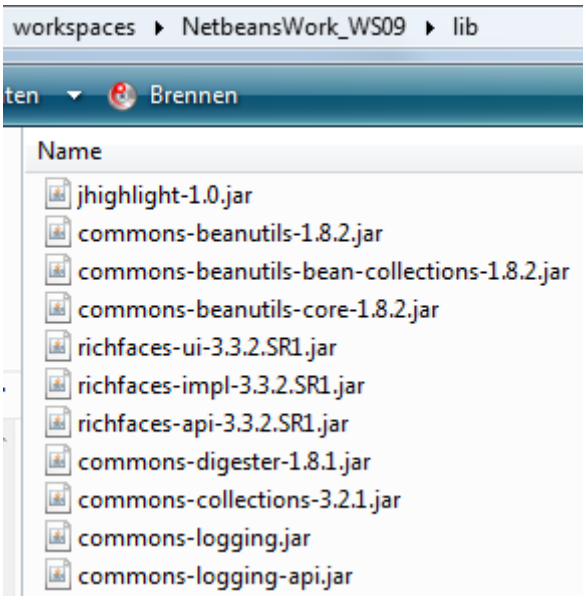
Zunächst benötigt man die Implementierung von Richfaces, die man über die Seite http://www.jboss.org/richfaces/download.html erhält. Dabei sollte ein „Stable“-Download genutzt werden, z. B. http://labs.jboss.com/downloading/?projectId=jbossrichfaces&url=http://download.jboss.com/jboss-

richfaces/richfaces-ui-3.3.2.SR1-bin.zip.

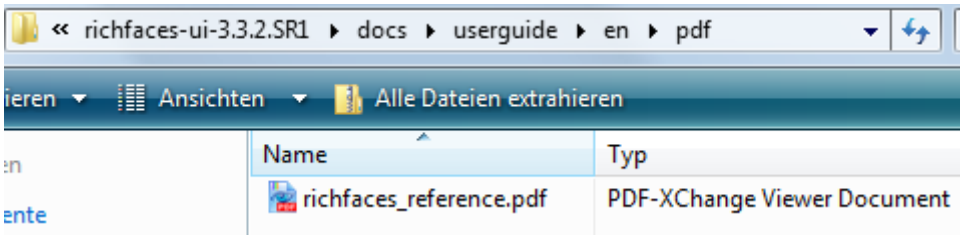
Innerhalb des Zip-Files existiert ein lib-Verzeichnis. Die drei enthaltenen jar-Dateien werden in ein Verzeichnis auf der Festplatte kopiert, z. B. ein lib-Verzeichnis auf der gleichen Höhe, wie die Projektverzeichnisse der NetBeans-Projekte.



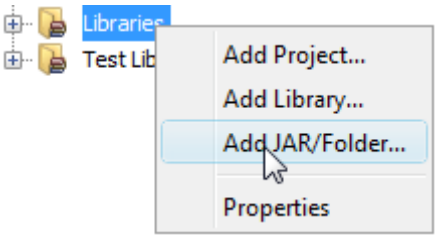
Weiterhin werden einige weitere Bibliotheken als jar-Dateien benötigt. Man muss beachten, dass die Versionsnummern zu den geforderten Versionsnummern passen, dabei können neue Versionen leider „zu neu“ und damit nicht kompatibel mit den alten Versionen sein. Für die hier verwandte RichFaces-Version werden folgende Bibliotheken benötigt, die man sich recht einfach zusammen googlen kann. Diese jar-Dateien werden in das gleiche Verzeichnis, wie die jar-Dateien von Richfaces gelegt.



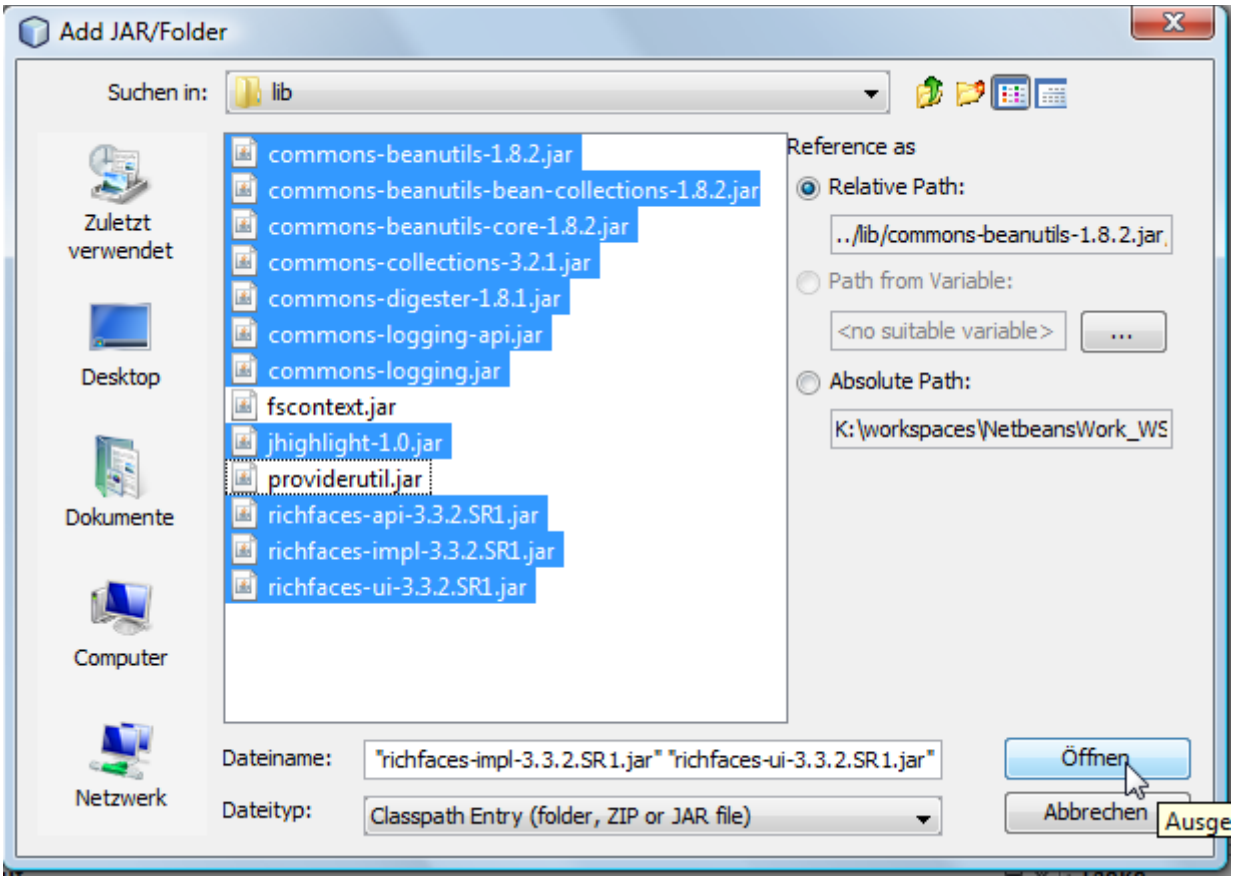
Neben einer Detaildokumentation enthält die RichFaces-Zip-Datei auch eine interessante einführende Dokumentation, in der auch die Komponenten von RichFaces allerdings recht knapp vorgestellt werden.



Es wird dann zunächst ein JSF-Projekt, wie am Anfang des Kapitels beschrieben, angelegt. Danach werden die jar-Dateien ergänzt. Dies geschieht durch einen Rechtsklick auf den Projektordner „Libraries“ in dem dann „Add JAR/Folder...“ ausgewählt wird.



Jetzt wird zum Verzeichnis mit den jar-Dateien manövriert und die angegebenen Dateien markiert. Arbeitet man z. B. auf einem USB-Stick empfiehlt sich rechts oben „Relative Path“ auszuwählen.



In der Datei web.xml müssen zumindest folgende Zeilen ergänzt werden.

```
<filter>
   <display-name>RichFaces Filter</display-name>
   <filter-name>richfaces</filter-name>
   <filter-class>org.ajax4jsf.Filter</filter-class>
 </filter>
 <filter-mapping>
   <filter-name>richfaces</filter-name>
   <servlet-name>Faces Servlet</servlet-name>
   <dispatcher>REQUEST</dispatcher>
   <dispatcher>FORWARD</dispatcher>
   <dispatcher>INCLUDE</dispatcher>
 </filter-mapping>
```

Danach kann RichFaces in den JSF-Dateien genutzt werden, z. B.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="f" uri="http://xmlns.jcp.org/jsf/core"%>
<%@taglib prefix="h" uri="http://xmlns.jcp.org/jsf/html"%>
<%@taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
<%@taglib uri="http://richfaces.org/rich" prefix="rich"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

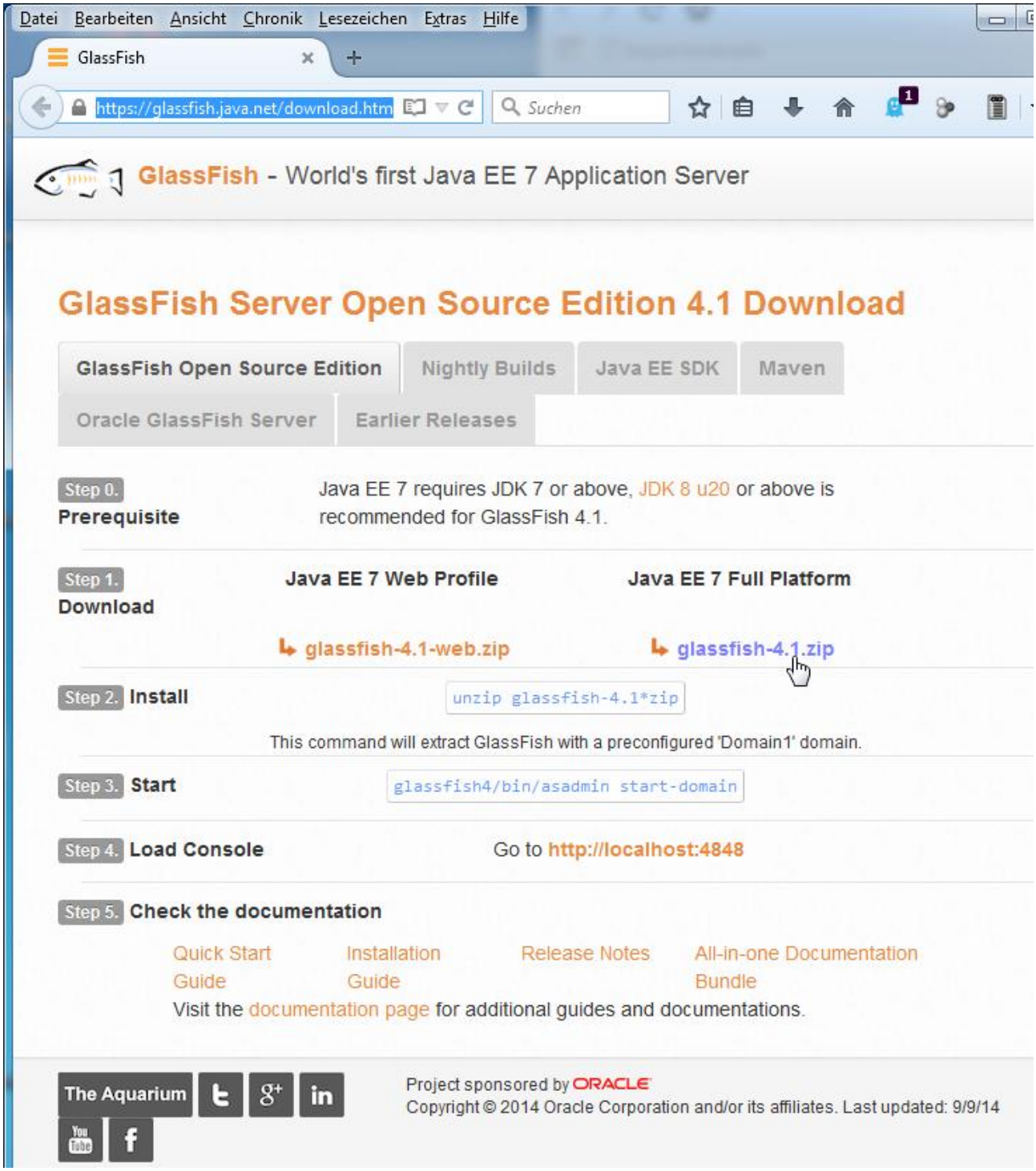
  <html>
    <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
      <title>RichFaces Spielerei</title>
    </head>
    <body>
      <f:view>
        <a4j:form>
          <h:inputText value="#{modul.name}">
            <a4j:support event="onkeyup" reRender="test"
                         limitToList="true"/>
          </h:inputText> <br>
          <h:outputText id="test" value="#{modul.name}" />
        </a4j:form>
      </f:view>
    </body>
  </html>
```

10.12 Installation eines neuen Servers

NetBeans kann mit verschiedenen Servern zusammenarbeiten, die theoretisch sogar parallel genutzt werden können, wenn sie unterschiedliche Portnummern nutzen. Hilfreich ist dieser Ansatz auch, wenn man einen neuen Server z. B. für eine portable Installation ergänzen möchte. Dies kann ebenfalls sinnvoll sein, wenn die Vermutung naheliegt, dass es Probleme im Server gibt, wenn z. B. die Sicherheitseinstellungen „verkonfiguriert“ wurden.

Die folgende Anleitung ist auch für den Server Payara (https://www.payara.fish/) nutzbar, der auf GlassFish basiert, diesen aber etwas optimiert und teilweise kleine Fehler beseitigt. Bei der nachfolgenden Installation wird Payara einfach mit den GlassFish-Einstellungen genutzt.

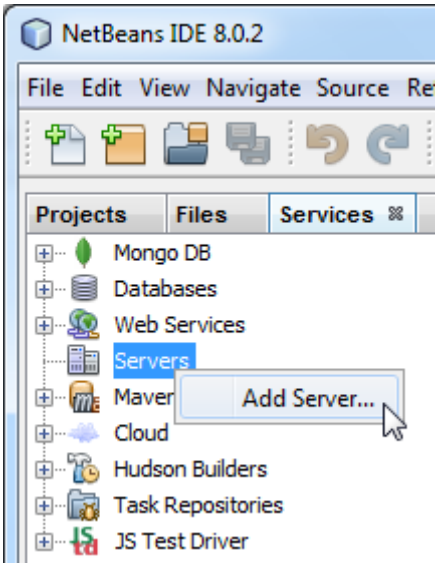
GlassFish kann direkt von der Seite https://glassfish.java.net/download.html heruntergeladen werden, es wird die Zip-Version hier genutzt.



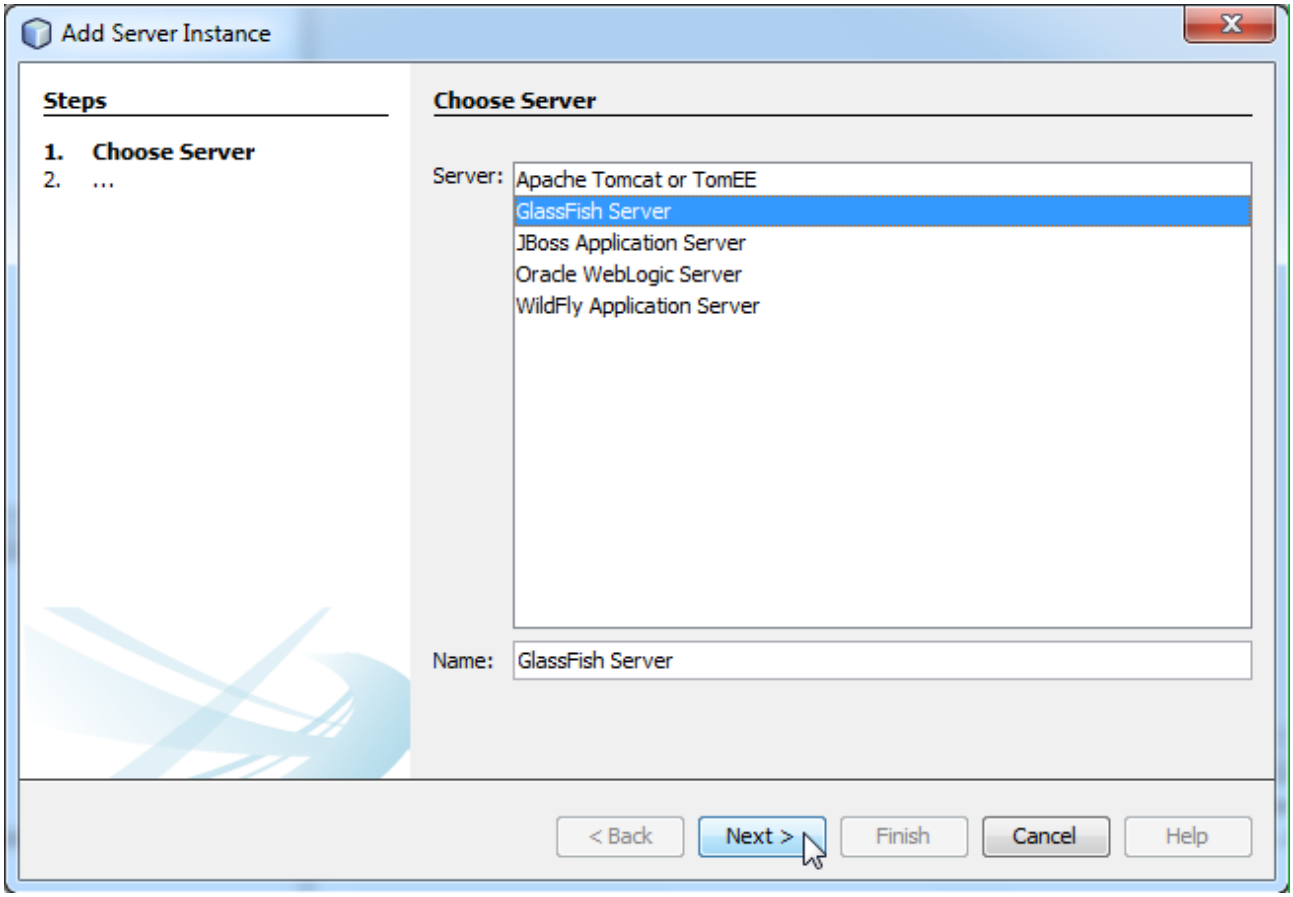
Die Zip-Datei kann in einem beliebigen, les- und schreibbaren Ordner entpackt werden. Historisch

wird meist angemerkt, dass der resultierende Pfad keine Leerzeichen enthalten soll, was aktuell nicht geprüft wurde.

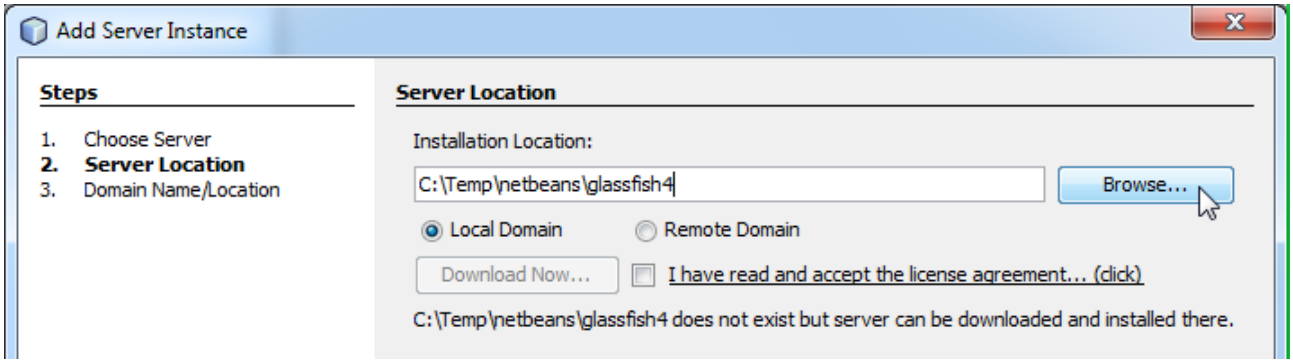
In NetBeans wird unter „Services“ ein Rechtsklick auf „Servers“ gemacht und „Add Server…“ ausgewählt.



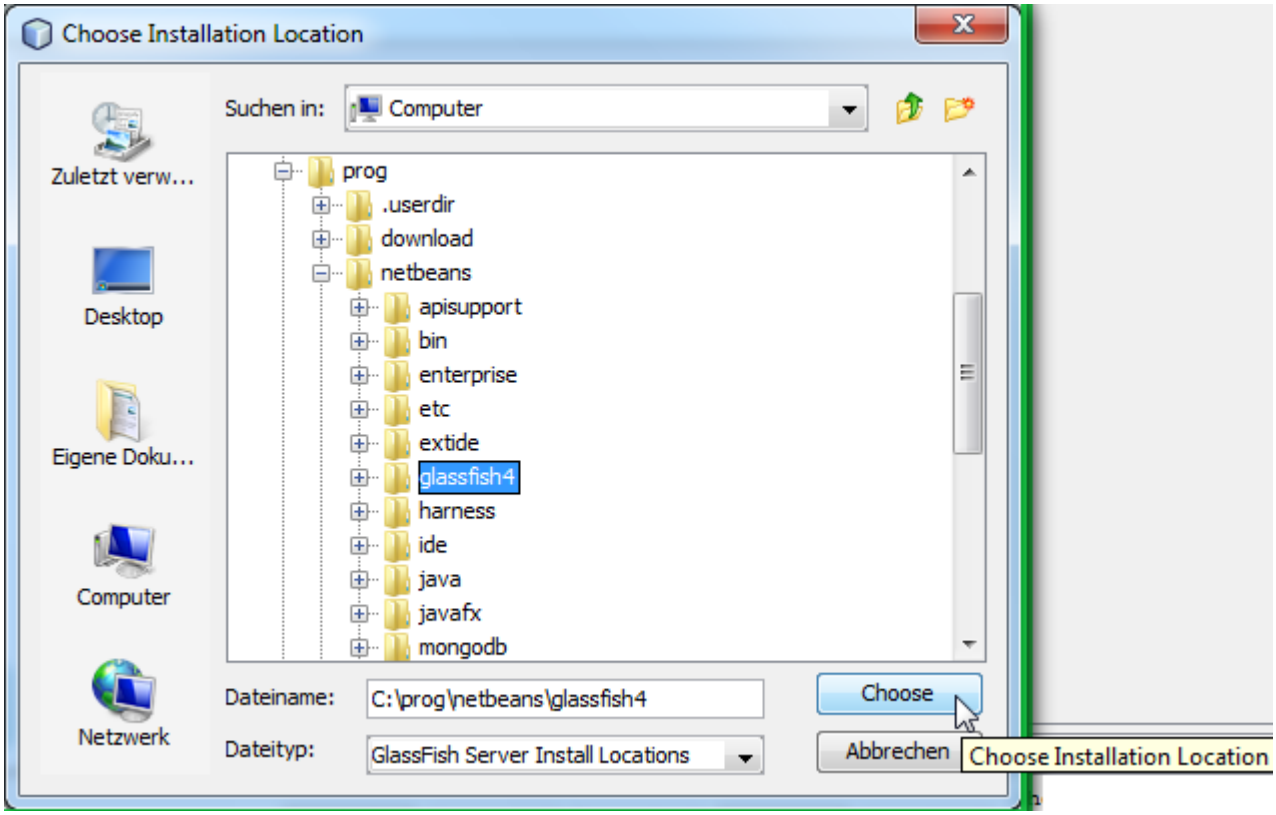
Es wird der passende Server gewählt und „Next >“ geklickt.



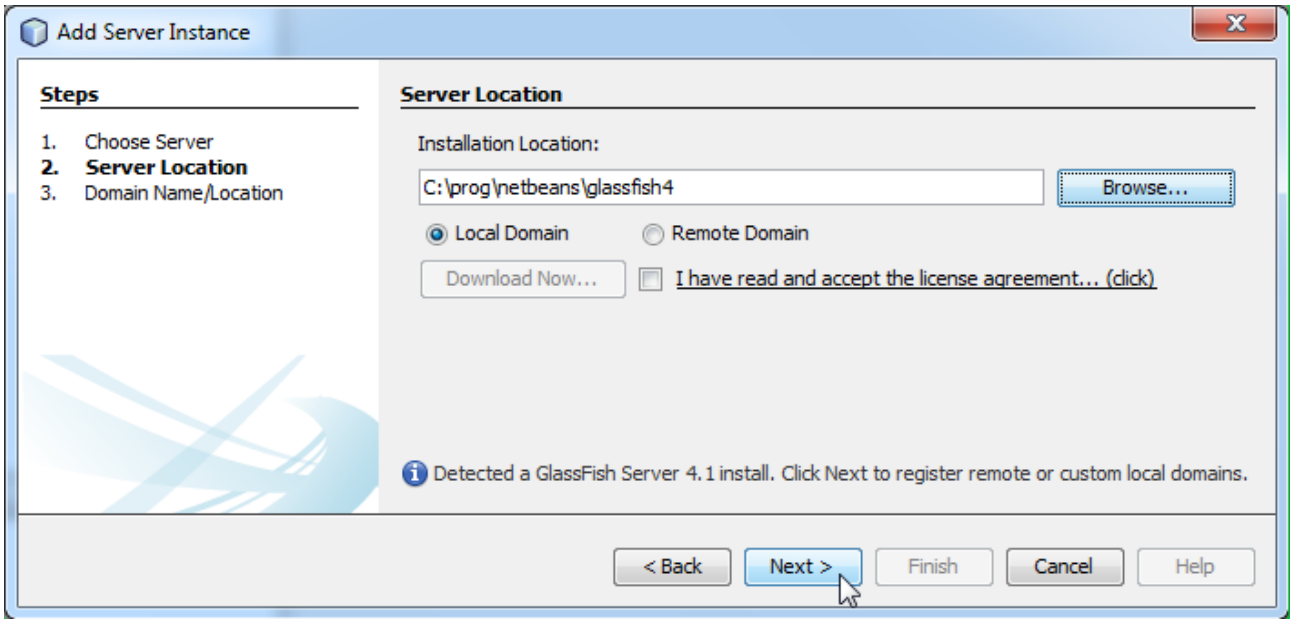
Über den rechten „Browse…“-Knopf wird jetzt zum GlassFish-Verzeichnis manövriert,



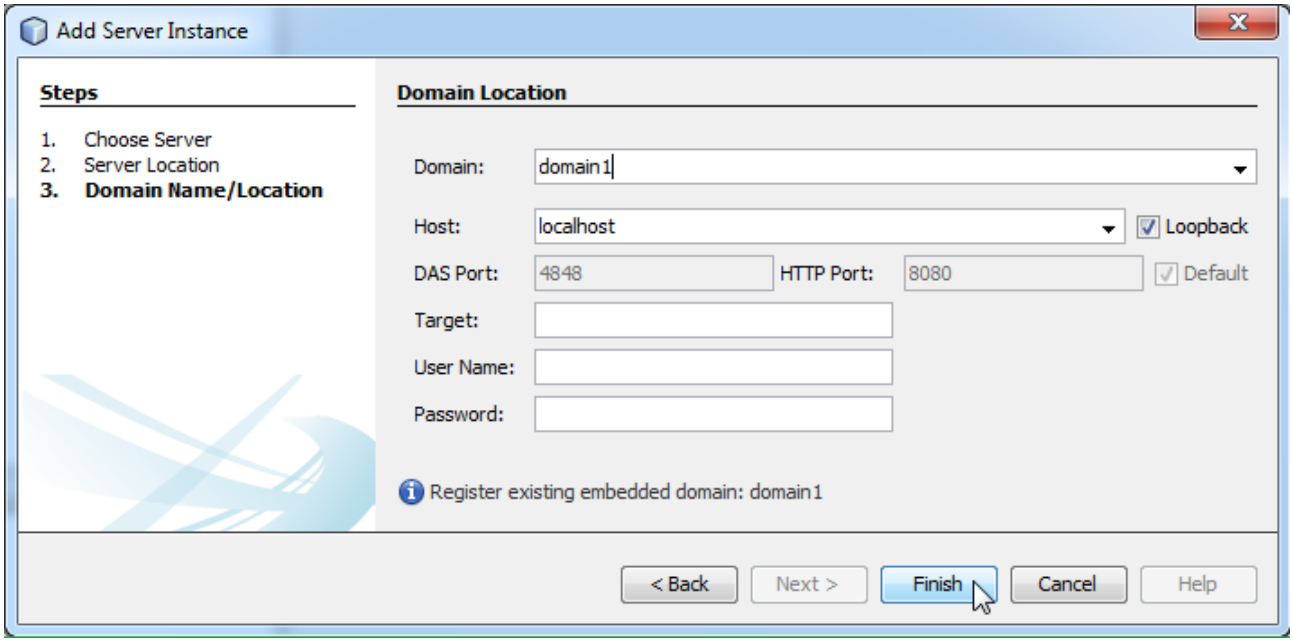
Es wird dabei nicht in dieses Verzeichnis hinein gegangen sondern direkt „Choose“ geklickt.



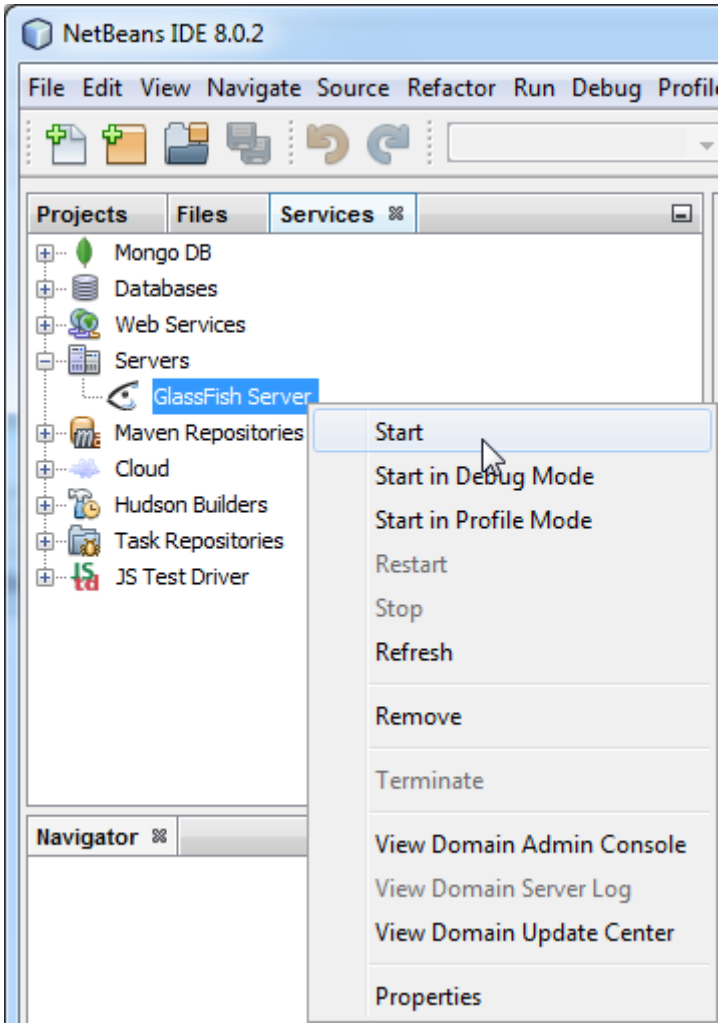
GlassFish wird automatisch gefunden, was durch die untere Textzeile angemerkt wird. Es wird „Next >“ geklickt.



Die Einstellungen können für Entwicklungszwecke übernommen werden. Es wird zum Abschluss der Integration von GlassFish in NetBeans „Finish“ geklickt.



Man erkennt, dass der Server existiert. Über einen Rechtsklick kann er gestartet werden.


NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profil
Projects
Files
Services
Mongo DB
Databases
Web Services
Servers
GlassFish Server
Maven Repositories
Cloud
Hudson Builders
Task Repositories
JS Test Driver
Navigator
Start
Start in Debug Mode
Start in Profile Mode
Restart
Stop
Refresh
Remove
Terminate
View Domain Admin Console
View Domain Server Log
View Domain Update Center
Properties

11 JavaFX (NetBeans 8.1)

JavaFX löst Swing als Framework zur Entwicklung graphischer Oberflächen ab. Generell können JavaFX-Klassen wie andere Java-Klassen auch genutzt werden. Eine Besonderheit besteht darin, wie die zentrale Klasse Application vom Class-Loader behandelt wird. Die Klasse ist immer der zentrale Ausgangspunkt eines Programmes, eine von Application erbende Klasse startet das Programm durch einen Aufruf mit Application.launch(args). Nur mit Tricks können Referenzen auf solche Objekte anlegt werden, der vorgegebene Ansatz fordert, dass genau eine von Application erbende Klasse für den Start und Aufbau des Programmes verantwortlich ist.

Verwandt zu Swing kann man in JavaFX Oberflächen strukturiert nach dem Schachtel-in-Schachtel-Prinzip programmieren. Dabei hat ein solcher Container ein zugewiesenes Layout und beinhaltet dann wieder weitere Container, die wieder ein Layout und noch weitere Container beinhalten können. Es entsteht so ein Baum von GUI-Komponenten, wobei jeder Knoten sehr individuell gestaltbar ist.

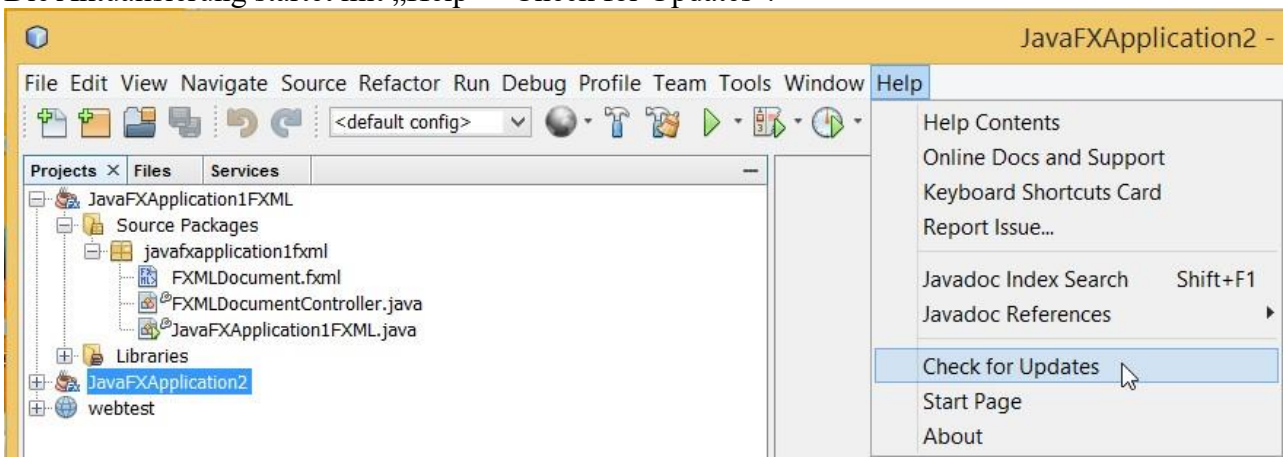
Neben der rein programmatischen Erstellung einer Oberfläche im Code, ermöglicht JavaFX auch die Gestaltung von Oberflächen mit FXML, einer speziellen XML-Variante. Für diese Variante seht ein Programm zur graphischen Entwicklung von Oberflächen mit dem Scene Builder zur Verfügung, der in NetBeans eingebunden oder völlig getrennt davon genutzt werden kann.

Die folgenden Unterkapitel zeigen die Integration von JavaFX in NetBeans, wobei generell auch mit einfachen Java-Projekten gearbeitet werden kann.

Bei einer Neuinstallation der Version 8.0.2 ist es sinnvoll, zuerst die im folgenden Kapitel allgemein vorgestellte Aktualisierung von NetBeans durchzuführen, da sonst die Integration von Scene Builder 8 in NetBeans 8.0.2 nicht möglich ist. Dieses Problem ist in NetBeans 8.1 behoben.

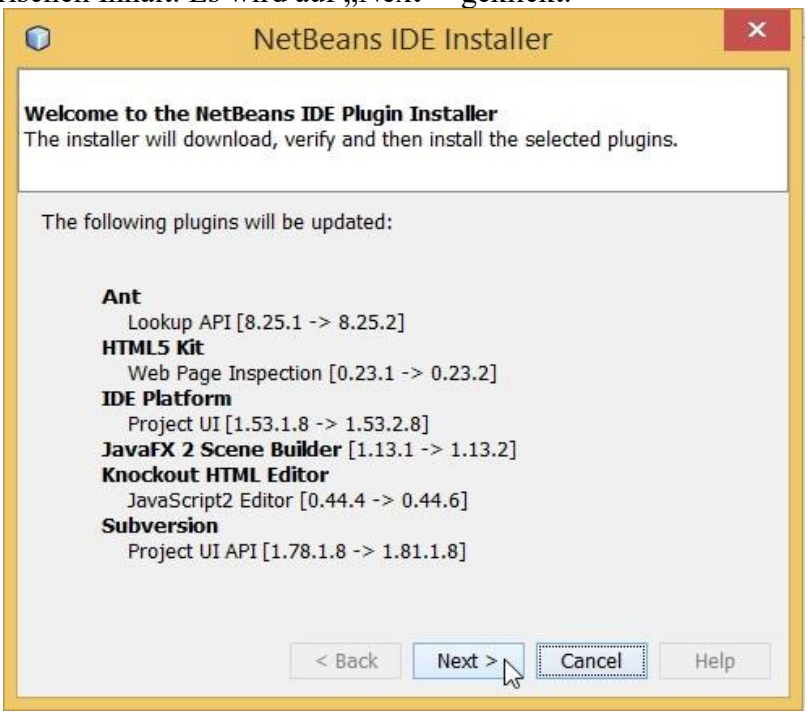
11.1 Aktualisierung von NetBeans

Nach dem Herunterladen und der Installation von NetBeans ist es sinnvoll, nach Aktualisierungen zu suchen. Ob man generell Aktualisierungen durchführen sollte, ist durchaus diskutabel, da es gerade bei sehr großen Projekten wichtig ist, dass alle Beteiligten eine einheitliche stabile Entwicklungsumgebung nutzen.
Die Aktualisierung startet mit „Help -> Check for Updates“.



Danach wird nach Aktualisierungen gesucht und diese angezeigt. Die folgende Abbildung zeigt dabei

nur einen exemplarischen Inhalt. Es wird auf „Next>“ geklickt.



Nachdem die Aktualisierungen geladen sind, wird in der rechten unteren Ecke eine Information angezeigt, die dazu auffordert, NetBeans zu schließen und wieder neu zu starten.



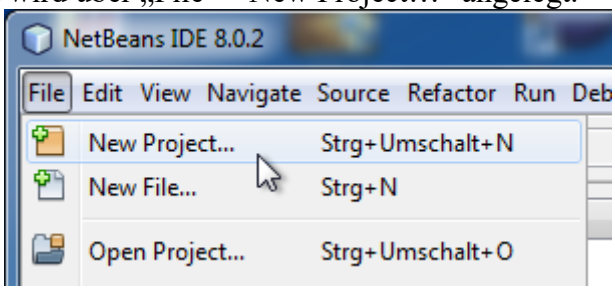
Beim Neustart ist dann ein spezielles Update-Fenster sichtbar, das die laufenden Installationen anzeigt.



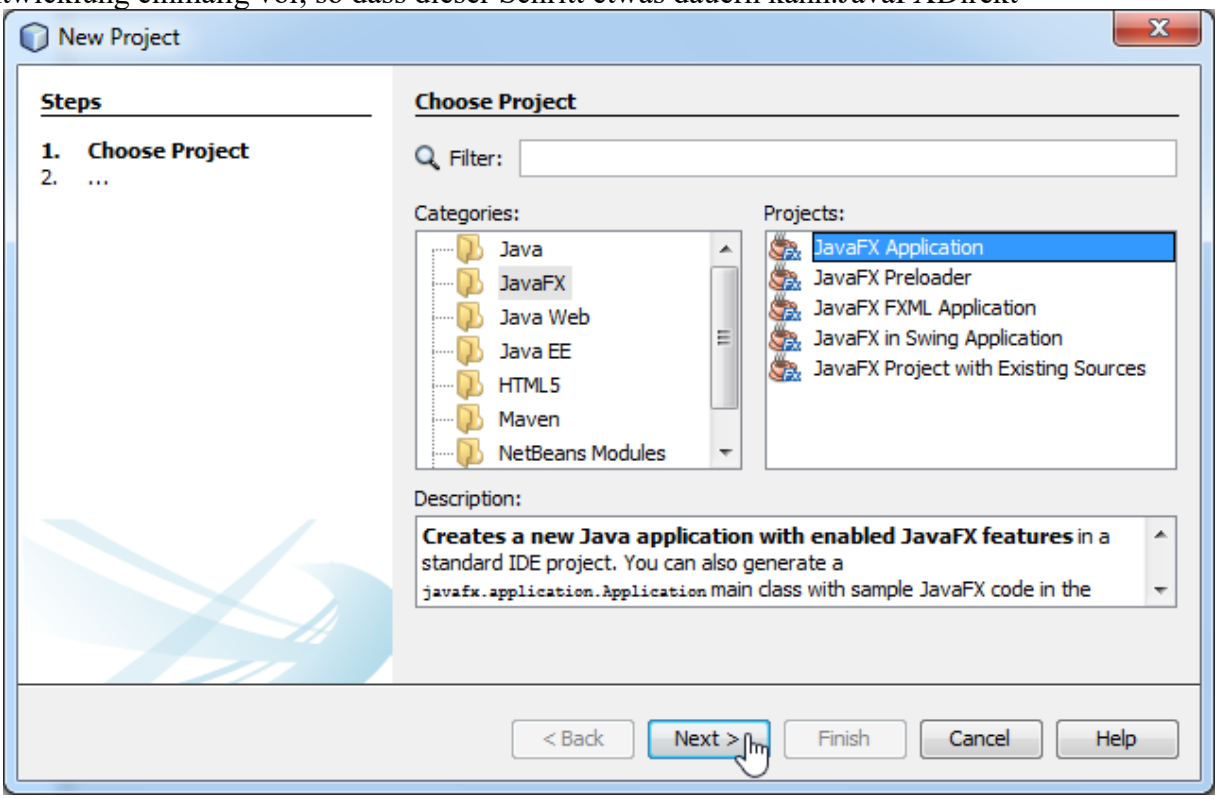
Danach kann mit NetBeans normal weitergearbeitet werden.

11.2 Erstellung eines JavaFX-Projekts

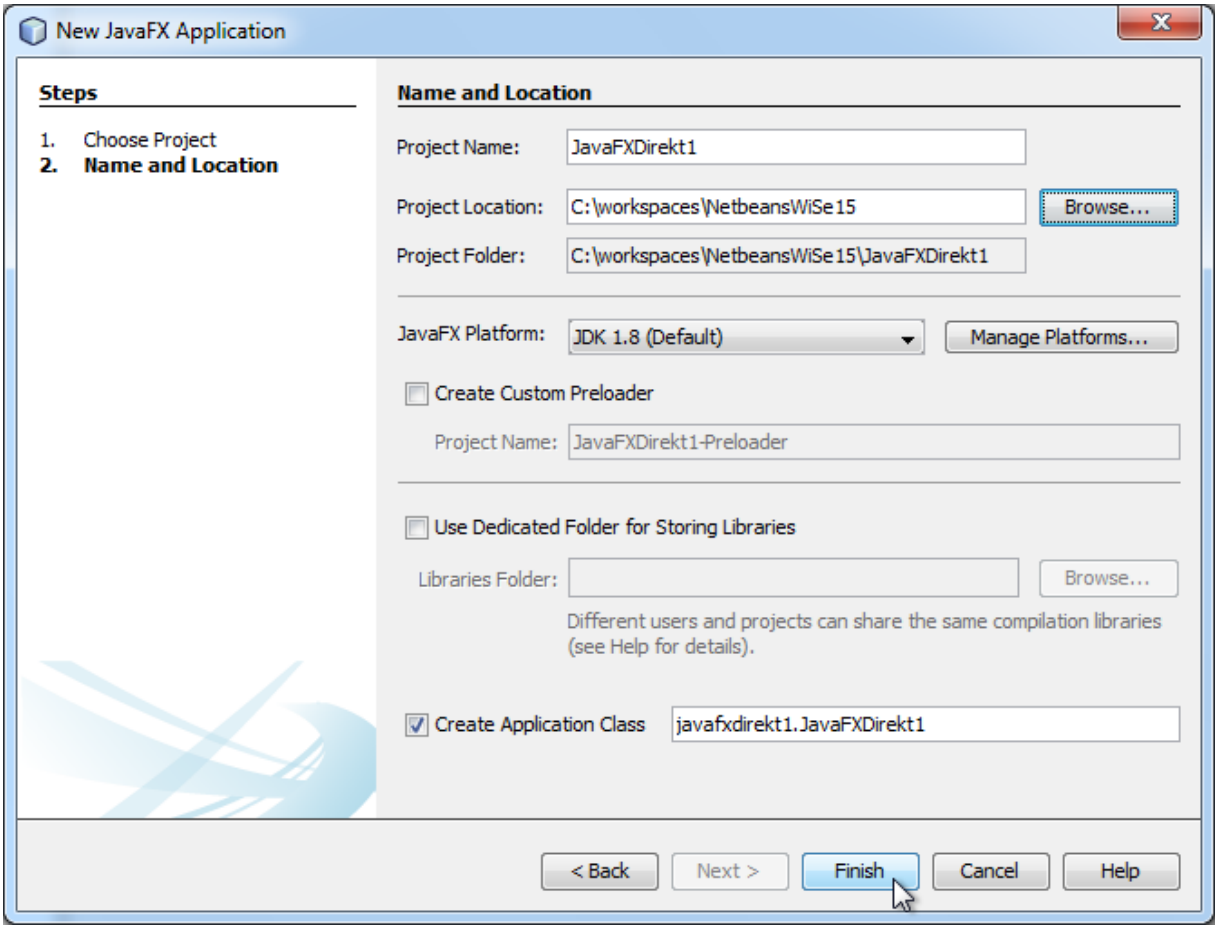
Generell kann ein JavaFX-Projekt wie ein Standard-Java-Projekt erstellt werden. Die Unterstützung in NetBeans ergänzt etwas Code, der die Erstellung gerade für unerfahrene Nutzer etwas erleichtert. Ein neues JavaFX-Projekt wird über „File -> New Project…“ angelegt.



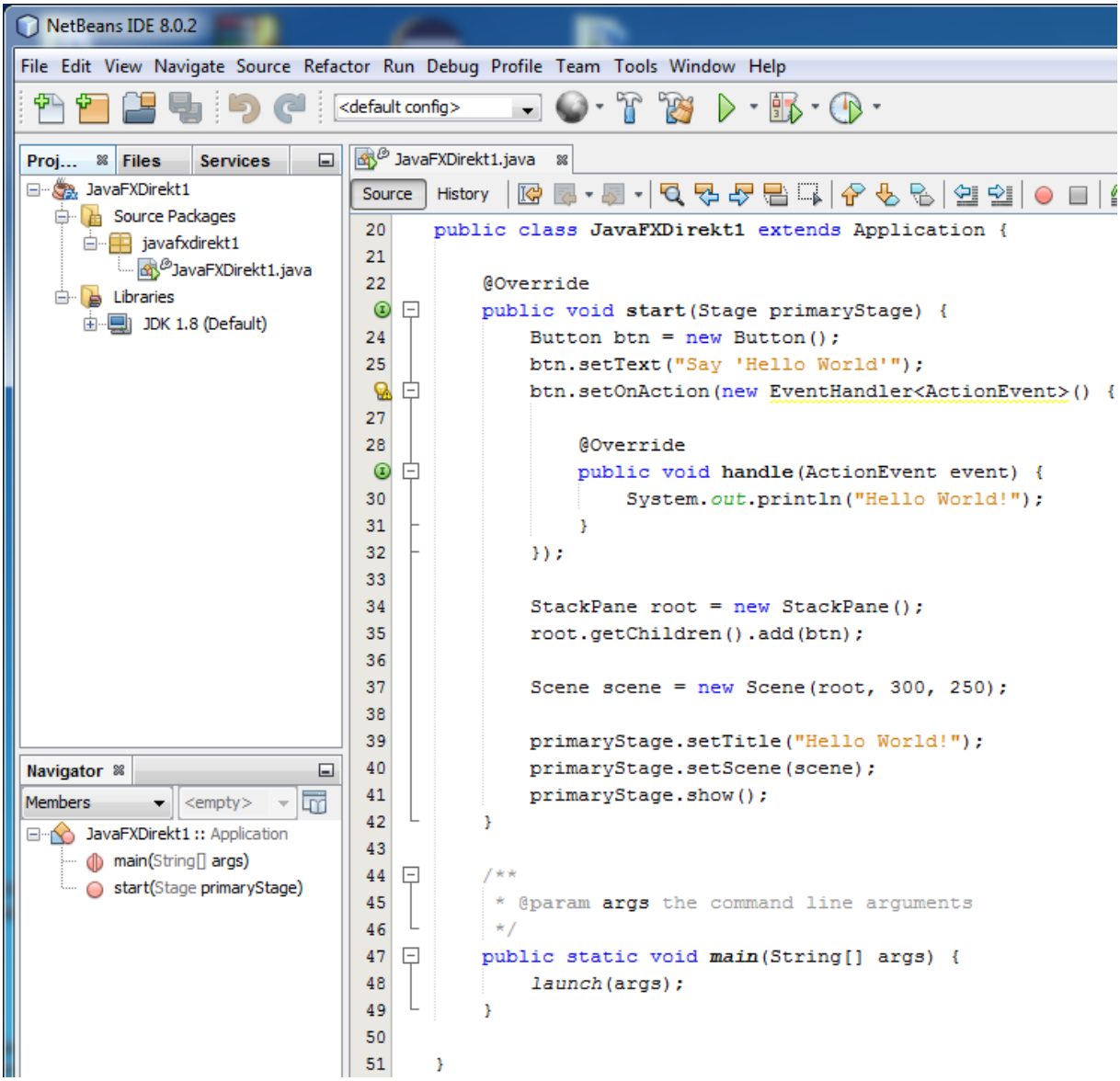
Auf der linken Seite wird „JavaFX“ ausgewählt. Auf der rechten Seite gibt es mehrere Vorschläge, welche Art von Projekt angelegt werden soll. Generell gilt allerdings, dass man durch die Programmierung den Stil später ändern kann. Typischerweise wird, wie hier, eine „JavaFX Application“ gewählt, wenn man die gesamte Oberfläche selbst Programmieren möchte. Sollte die Oberflächenentwicklung auf Basis von FML passieren, sollte „JavaFX FXML Application“ gewählt werden. Es wird auf „Next>“ geklickt. Bei der ersten Nutzung bereitet NetBeans die JavaFX-Entwicklung einmalig vor, so dass dieser Schritt etwas dauern kann.JavaFXDirekt



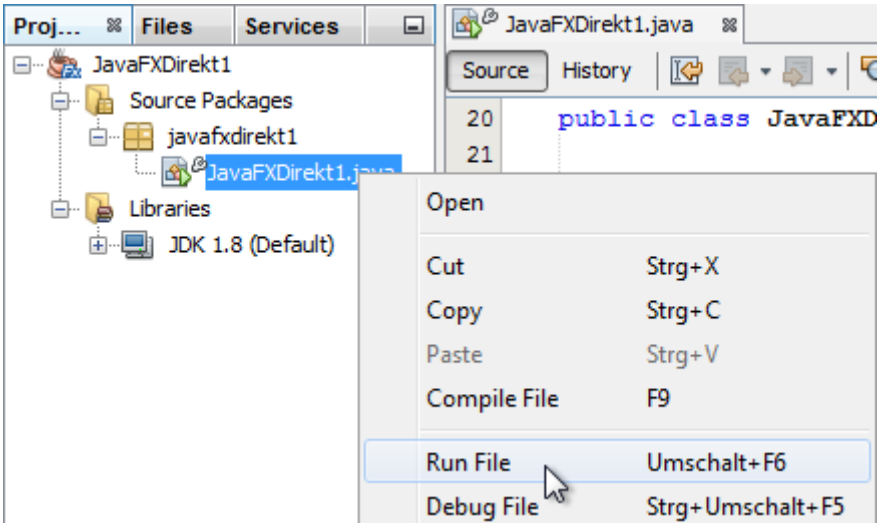
Oben wird dem Projekt ein Name gegeben, die weiteren Einstellungen kann man so lassen, unten wird der Name der zentralen Klasse festgelegt und „Finish“ geklickt.



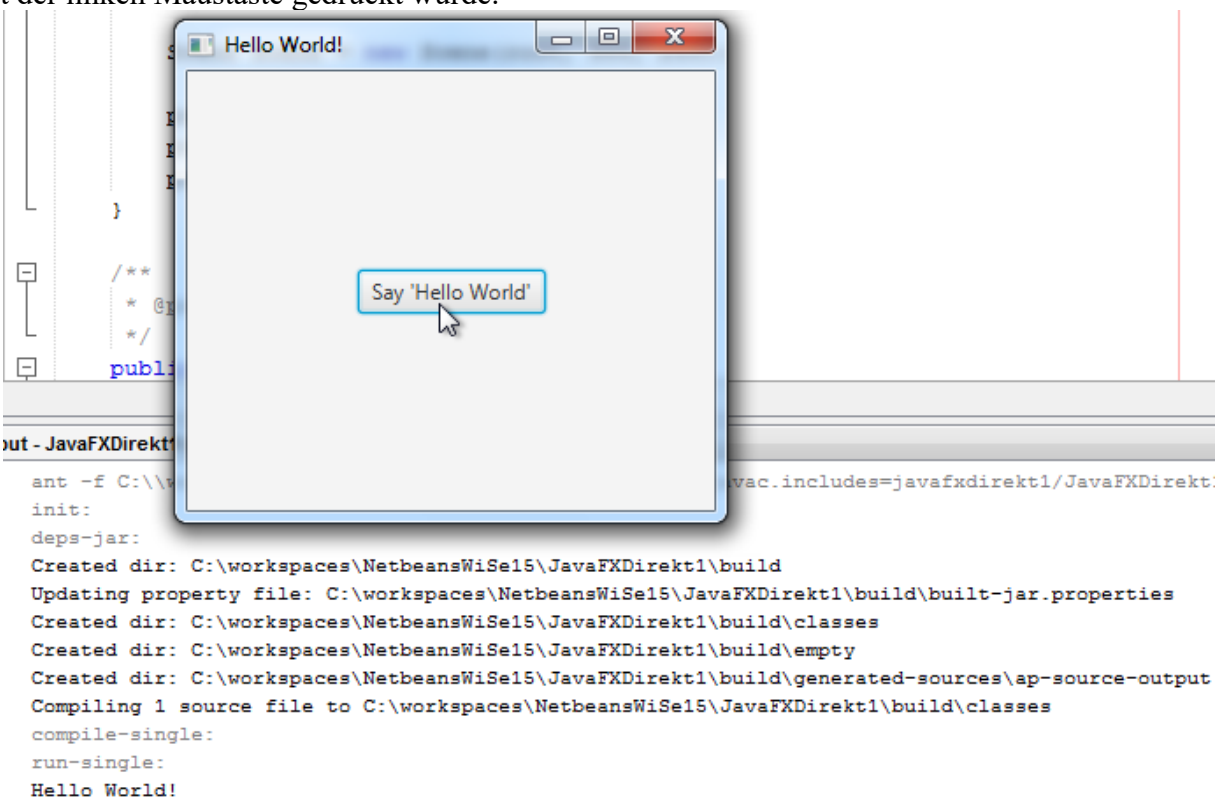
Es wird ein minimales Projekt mit Beispielcode angelegt, den man bei Bedarf auch einfach löschen kann.



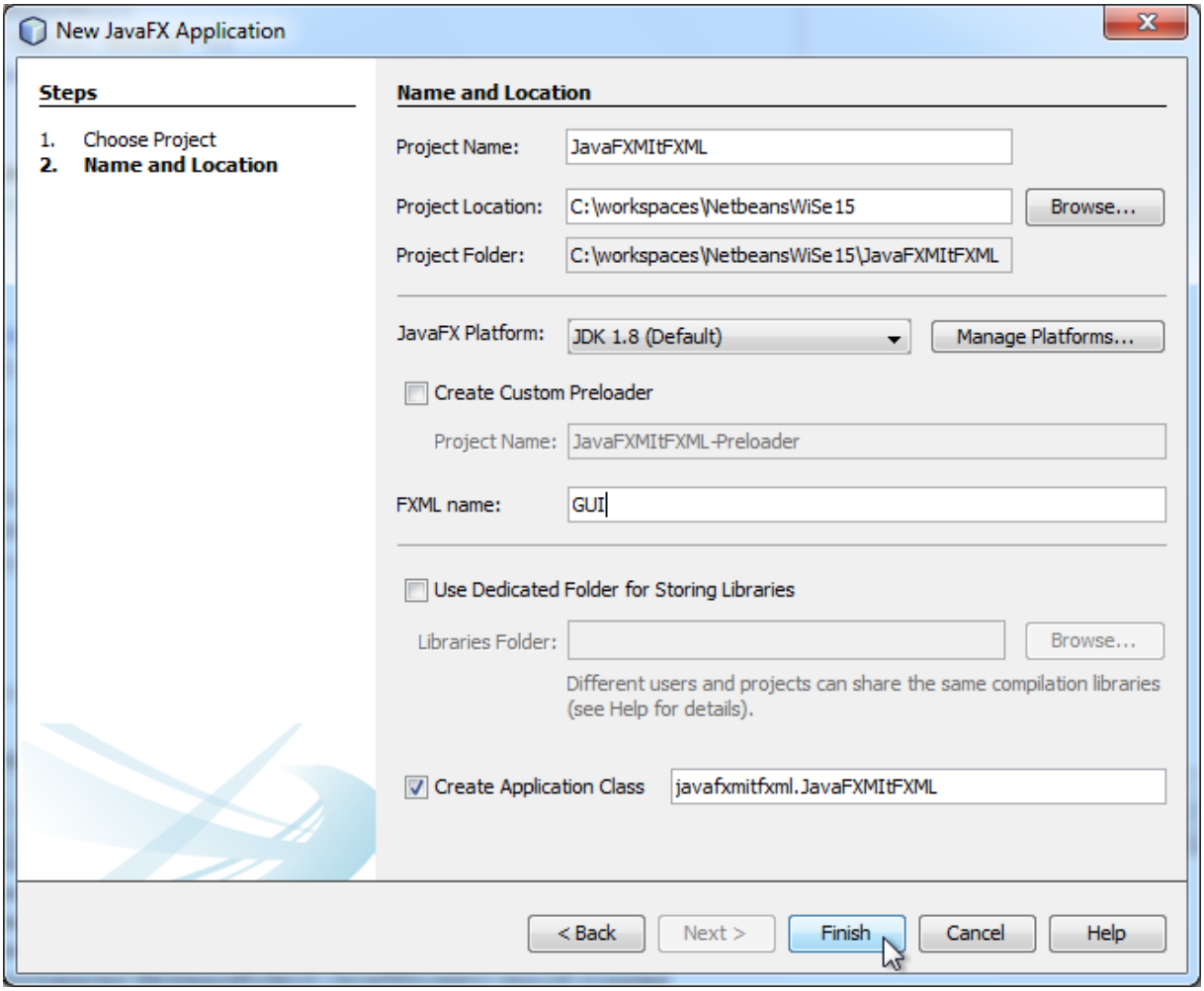
Die Applikation kann normal über den grünen Pfeil in der Kopfleiste oder über einen Rechtsklick auf der Datei gestartet werden.



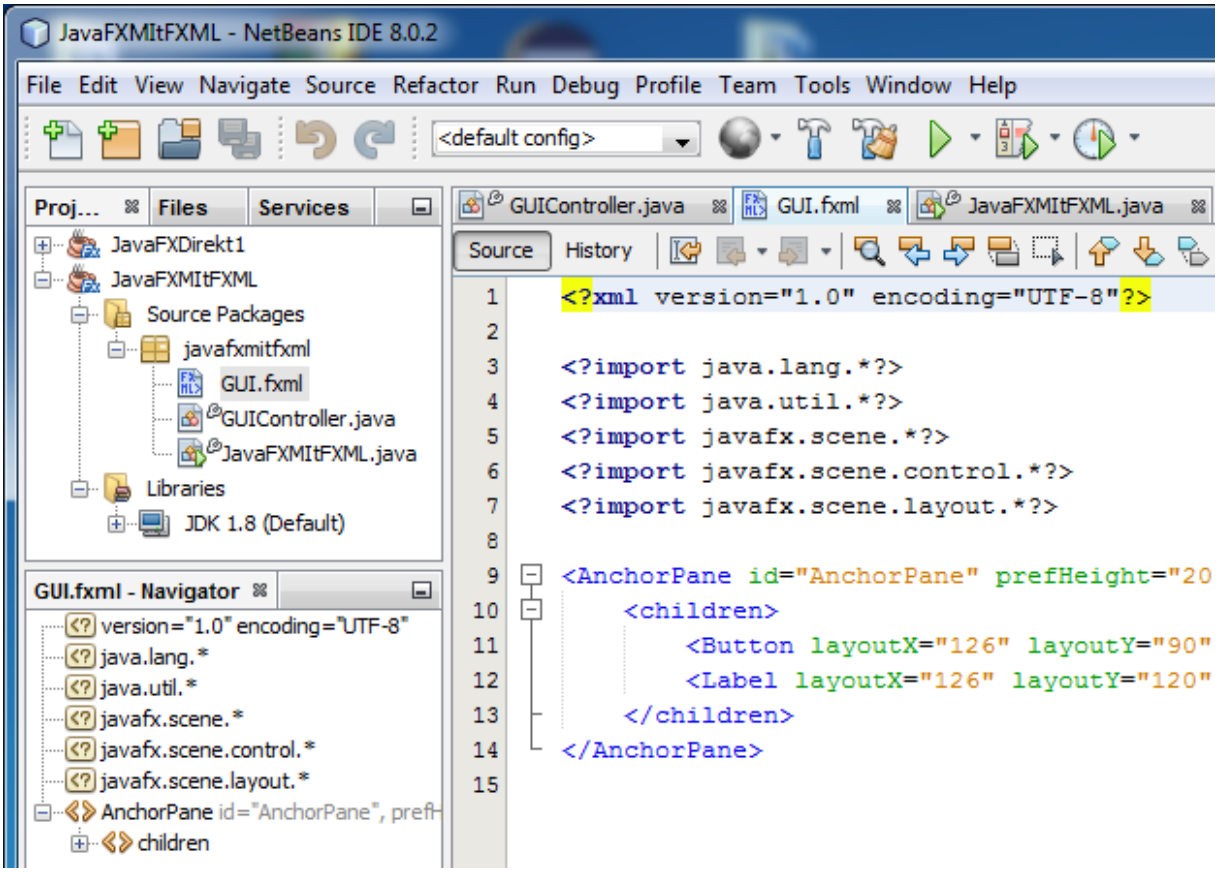
Das minimale Beispielprogramm erzeugt folgende Ausgabe nachdem der Knopf in der Mitte einmal mit der linken Maustaste gedrückt wurde.



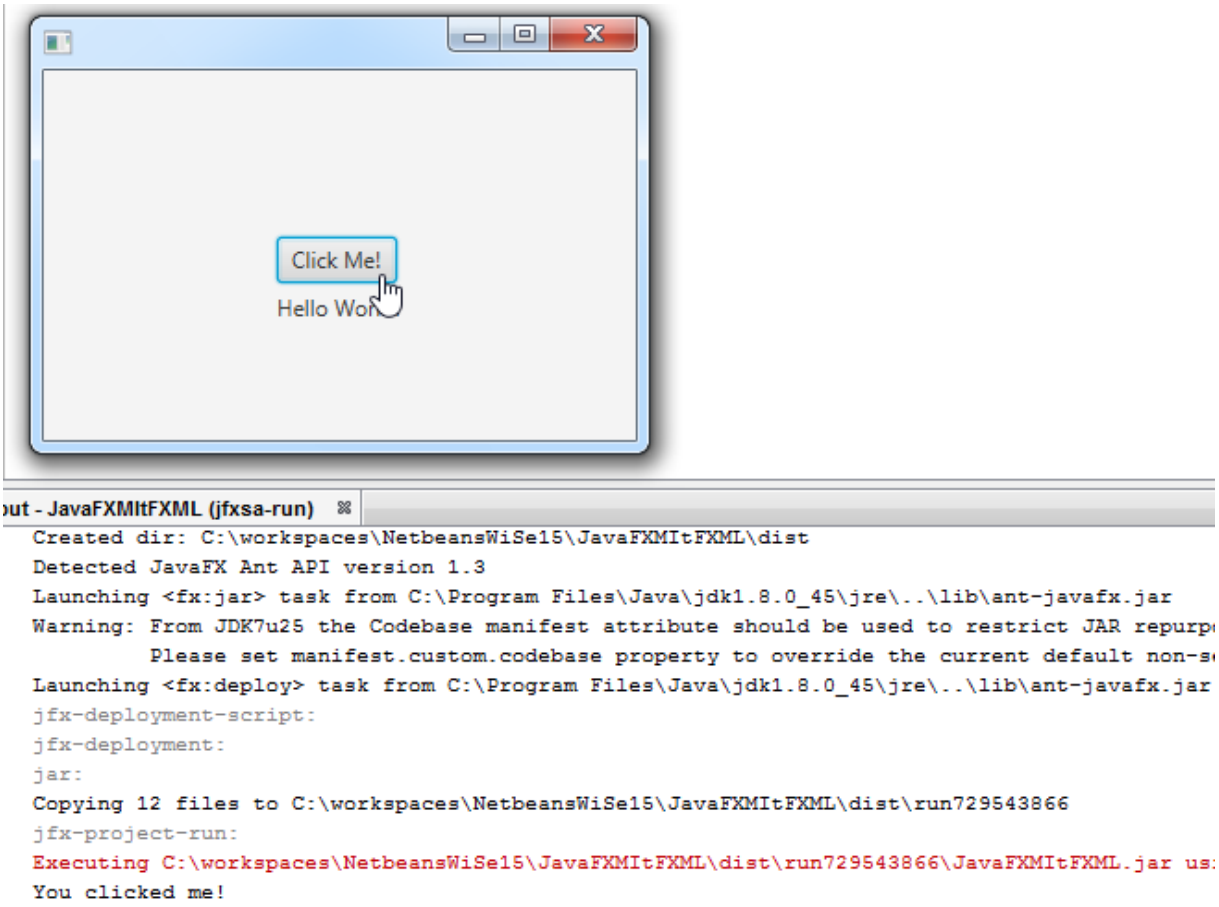
Bei der Auswahl eines FXML-Projekts kann bei der Projekterzeugung der Name der FMXL-Datei „FXML name“, hier GUI, angegeben werden.



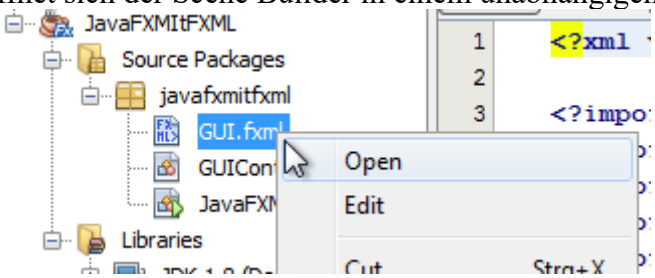
Es entstehen drei Dateien, die wieder eine Beispielapplikation erzeugen, allerdings auf FXML basieren.



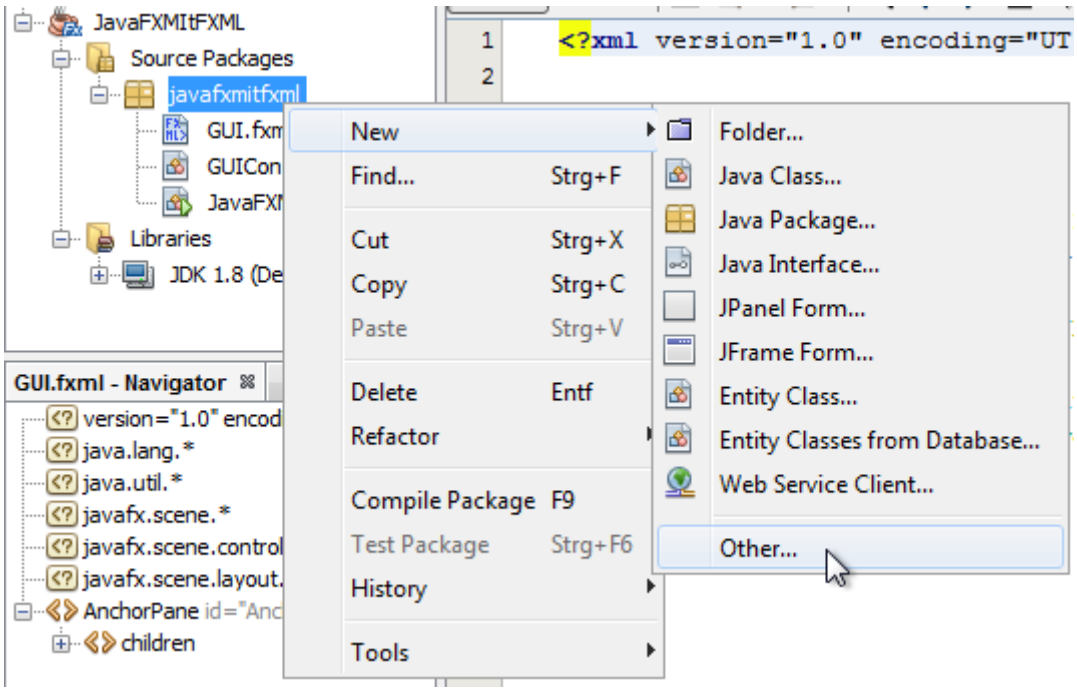
Die nachfolgende Abbildung zeigt die laufende Beispielapplikation nachdem der Knopf einmal mit der linken Maustaste geklickt wurde.



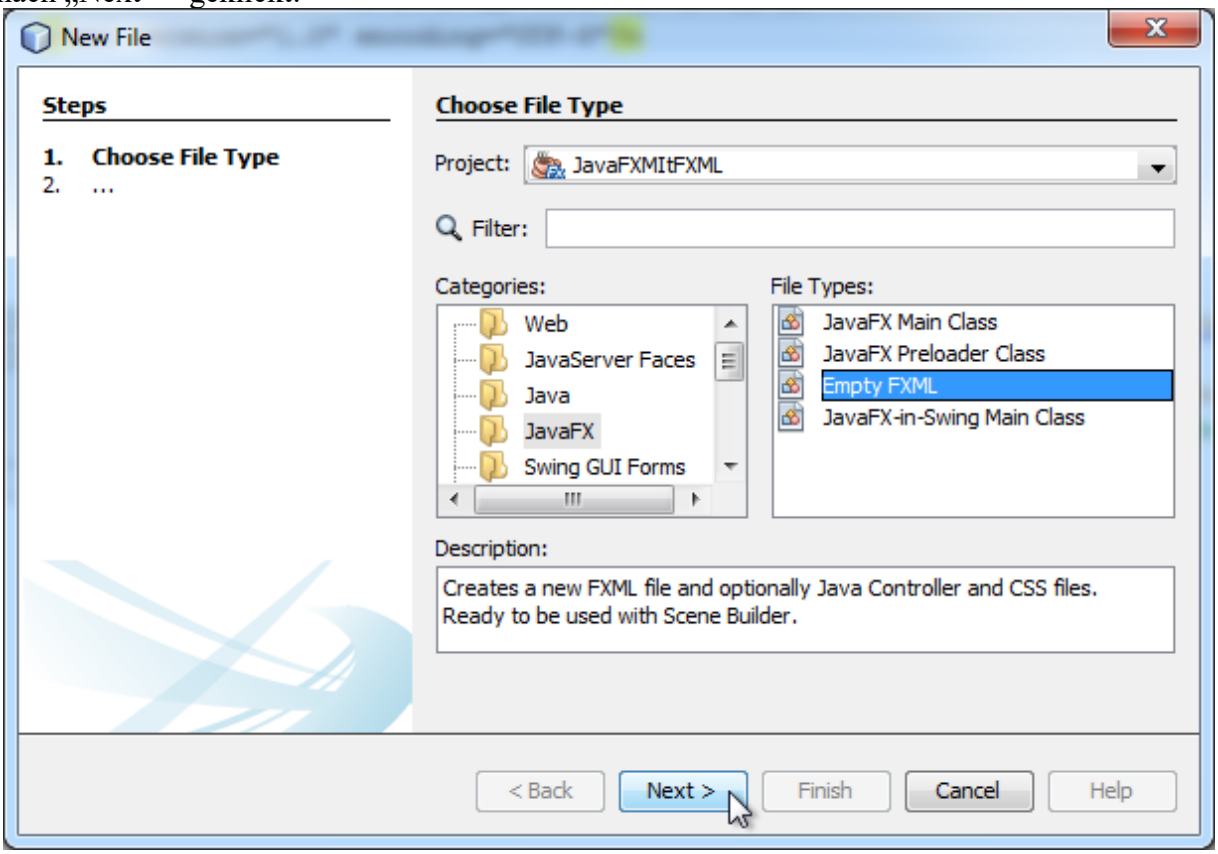
Sollte der Scene Builder, wie in „11.4 Installation des Scene Builders“ beschrieben, installiert sein, erhält man durch einen Rechtsklick auf einer FXML-Datei das Angebot, die Datei mit „Open“ oder „Edit“ zu bearbeiten. Bei der Standardoperation „Open“, die auch mit einem Doppelklick auf der Datei ausgeführt wird, öffnet sich der Scene Builder in einem unabhängigen Fenster.



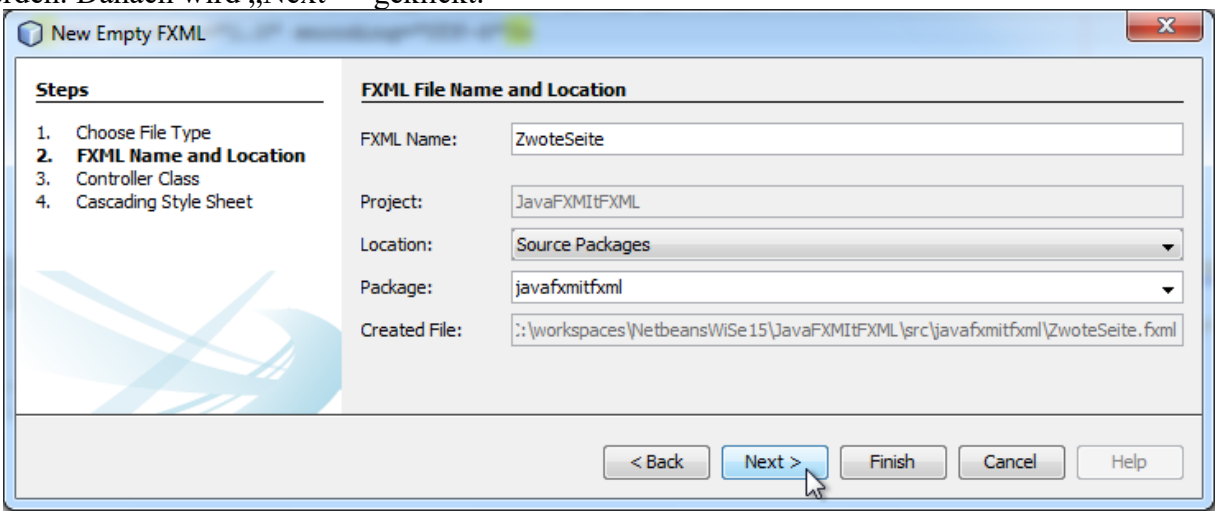
Soll eine neue FXML-Datei erstellt werden, kann z. B. ein Rechtsklick auf dem Paket gemacht werden, in dem die Datei angelegt werden soll. Danach wird „New“ angeklickt. Wird die FXML-Datei nicht direkt angeboten, wird unten zunächst „Other …“ ausgewählt.



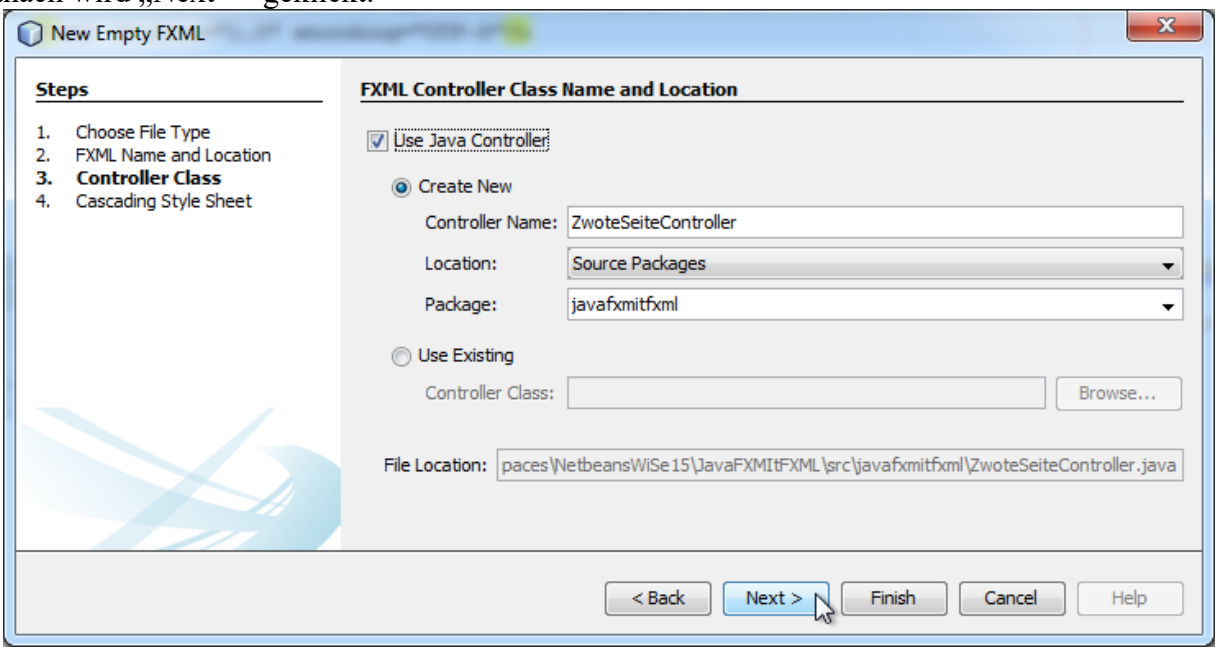
Mit der Wahl von „JavaFX“ auf der linken Seite, wird dann rechts „Empty FXML “ angeboten und danach „Next >“ geklickt.



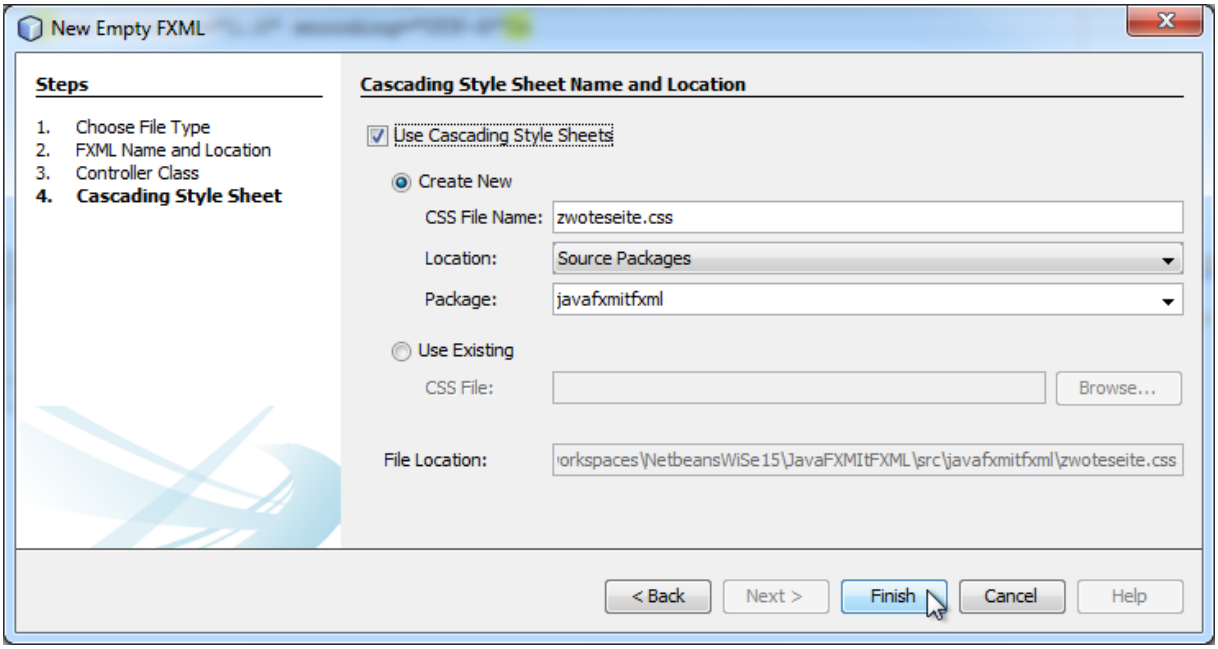
Dann wird oben der Name der Datei angegeben. Das genutzte Paket könnte auch hier eingestellt werden. Danach wird „Next >“ geklickt.



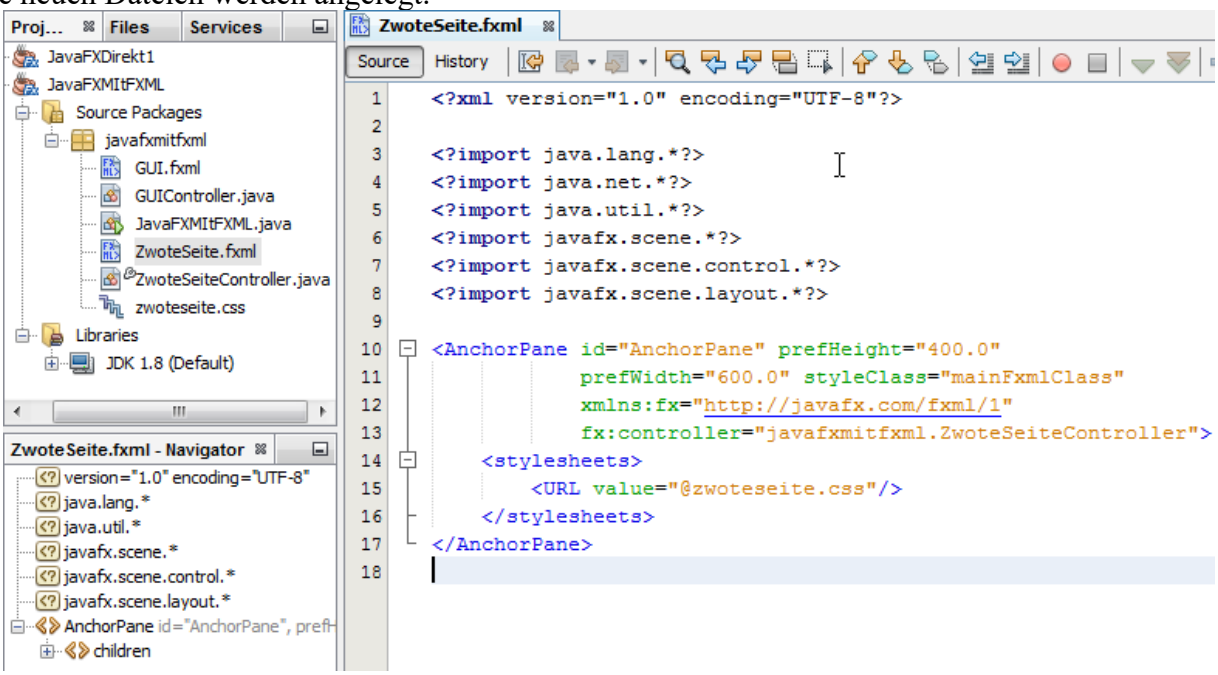
Es ist hier bereits möglich, eine Java-Controller-Klasse anlegen, dazu muss links-oben ein Haken gesetzt werden. Eine solche Datei kann man natürlich auch später in die FXML-Datei eintragen. Danach wird „Next >“ geklickt.



Man kann hier bereits eine CSS_Datei anlegen, dazu muss links-oben ein Haken gesetzt werden. Eine solche Datei kann man natürlich auch später in die FXML-Datei eintragen. Danach wird „Finish“ geklickt.



Die neuen Dateien werden angelegt.



11.3 JavaFX und NetBeans 8.0.2

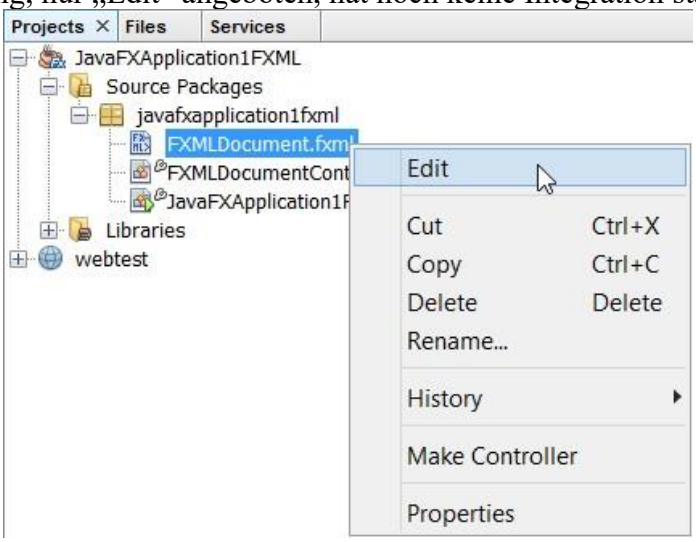
JavaFX folgt bestimmten Regeln bei der Übersetzung von Dateien, die teilweise auch Auswirkungen auf NetBeans haben, die zumindest in der Version 8.0.2 nicht ganz sauber berücksichtigt werden. Dies führt u. a. dazu, dass der Clean-And-Build-Schritt nicht sauber durchgeführt wird. Eine einfache Abhilfe ist es, dem Schließen-Icon einer JavaFX-Oberfläche explizit ein Schließen der Java Virtual

Maschine zuzuordnen. Konkret wird neben den sonstigen Schritten zum Aufräumen am Ende der letzte Befehl System.exit(0) für setOnCloseRequest(.) ergänzt.

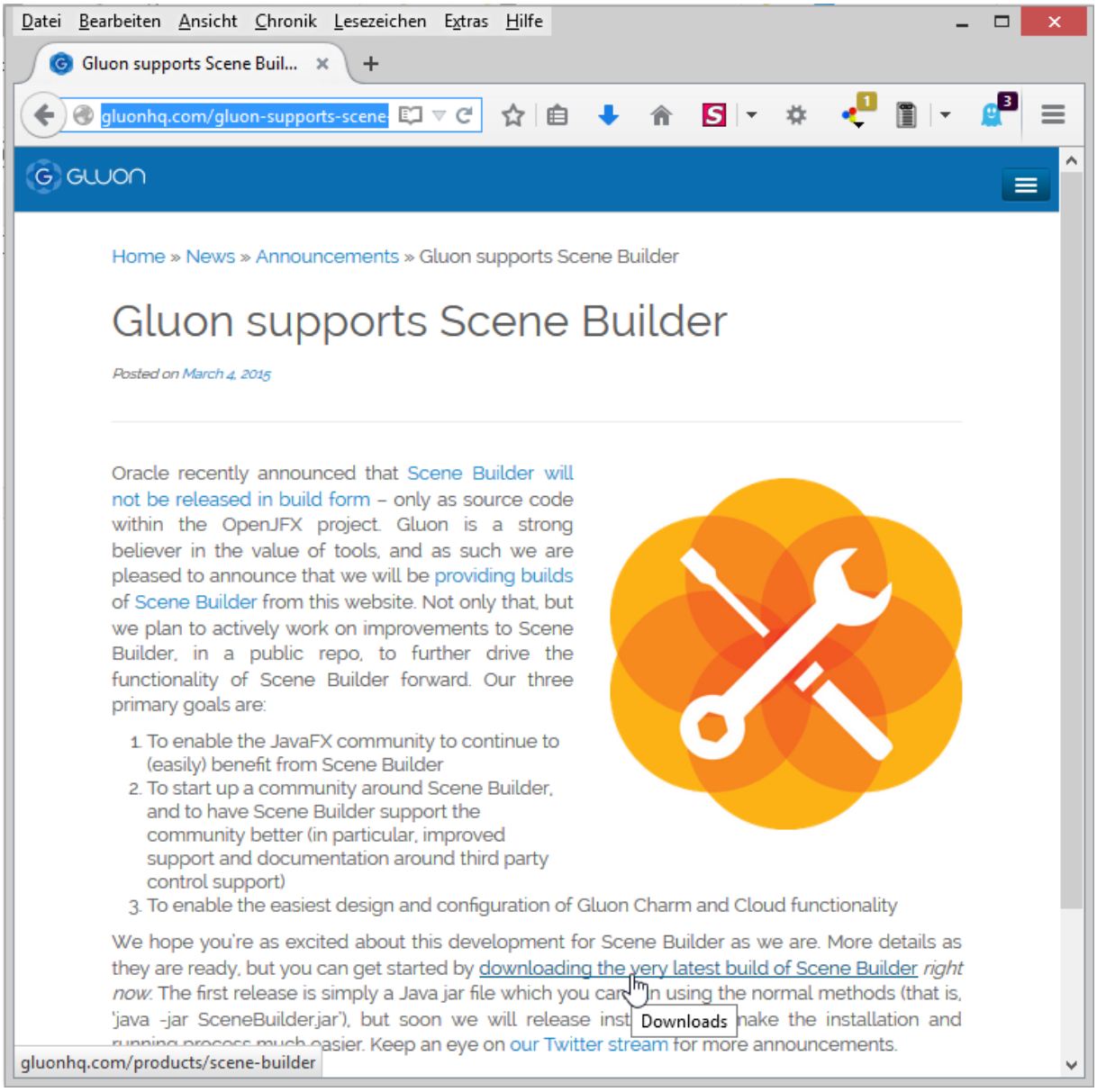
```
78              primaryStage.setScene(scene);
79              primaryStage.setOnCloseRequest(e -> System.exit(0));
80              primaryStage.show();
81          }
```

11.4 Installation des Scene Builders

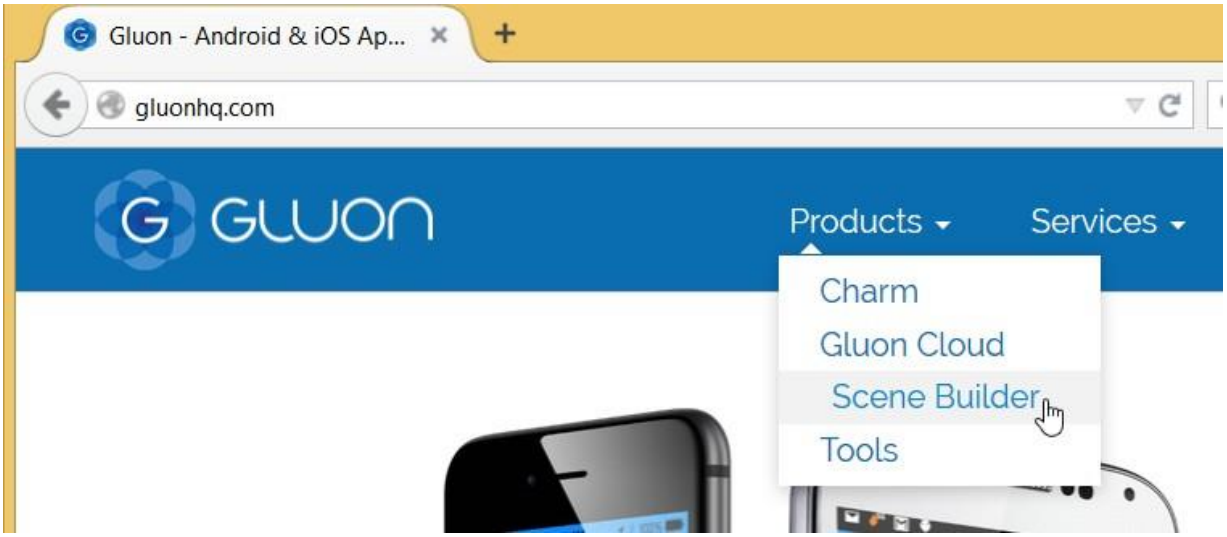
Generell kann der Scene Builder als Werkzeug zur Entwicklung einer JavaFX-Oberfläche auf Basis von FXML genutzt werden. Weiterhin ist eine Integration in NetBeans möglich. Ob diese Integration erfolgt ist, ist recht einfach durch einen Rechtsklick auf einer FXML-Datei ersichtlich. Wird, wie in der folgenden Abbildung, nur „Edit“ angeboten, hat noch keine Integration stattgefunden.



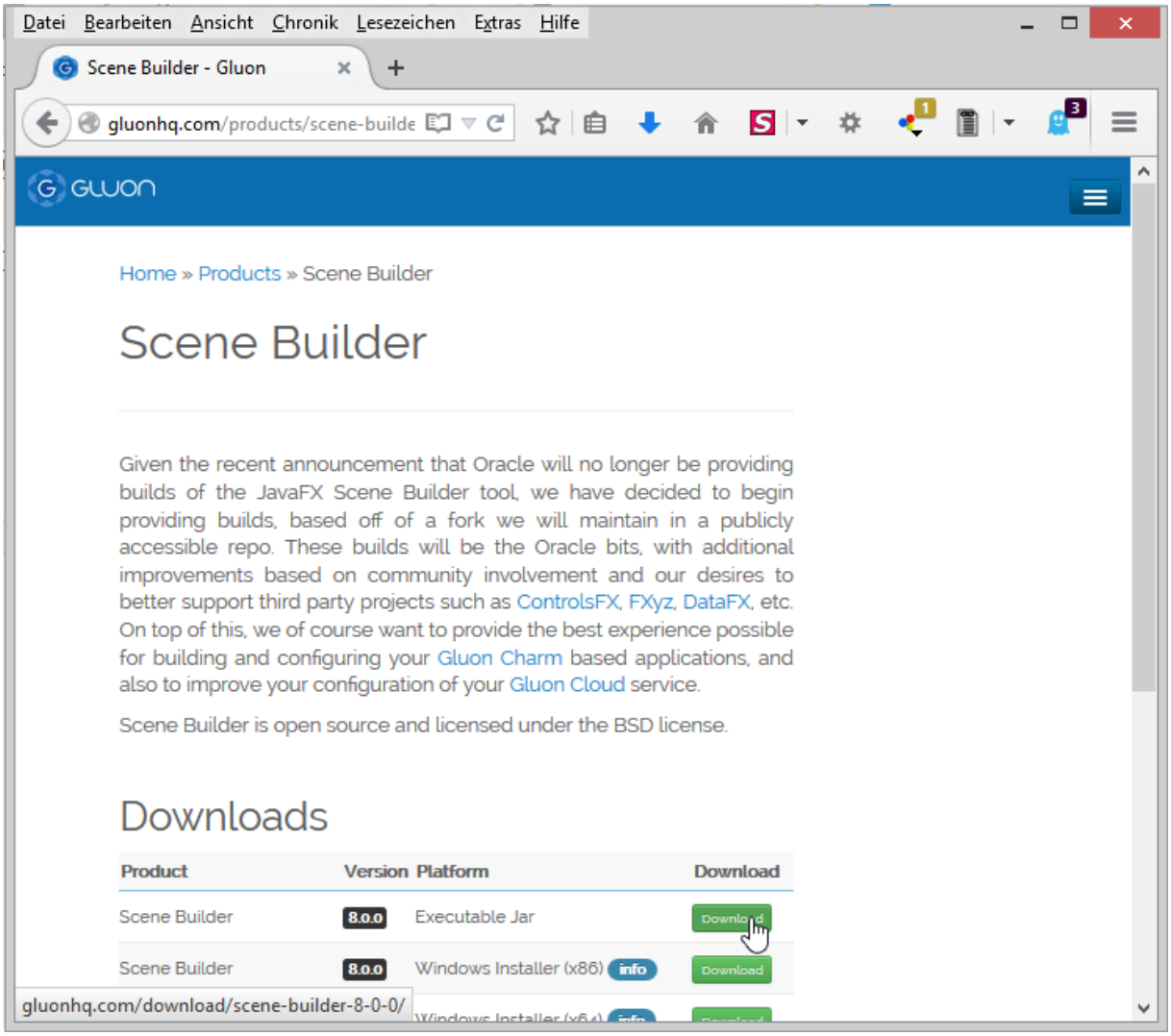
Ursprünglich und über längere Zeit wurde das Werkzeug von Oracle direkt zum Download angeboten. Seit der Java-Version 8.40 hat sich dies geändert. Dabei können rechtliche Probleme sowie der Wunsch, dass das Werkzeug von einer Community weiterentwickelt wird, eine Rolle gespielt haben. Oracle bietet die letzte gepflegte Version über die Seite http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html an, wobei diese nicht für aktuelle Java-Versionen selbst aktualisiert wird. Die angebotene Version ist für JavaFX 2 aber nicht mehr für die Nachfolgeversion JavaFX 8, wobei sich die Versionsnummer an die Java-Versionsnummer angepasst hat.
Auf der Seite http://www.oracle.com/technetwork/java/javase/downloads/sb2download-2177776.html verweist Oracle auf die OpenJDK-Community, wo der Builder geladen, dann aber selbst kompiliert werden muss. Die rechtlichen Rahmenbedingungen erlauben es, dass das Werkzeug in unveränderter, aber kompilierter Form auch von anderen angeboten werden kann. Ein solcher, hier genutzter, Anbieter ist Gluon (http://gluonhq.com/) auf der Web-Seite http://gluonhq.com/gluon-supports-scene-builder/ .



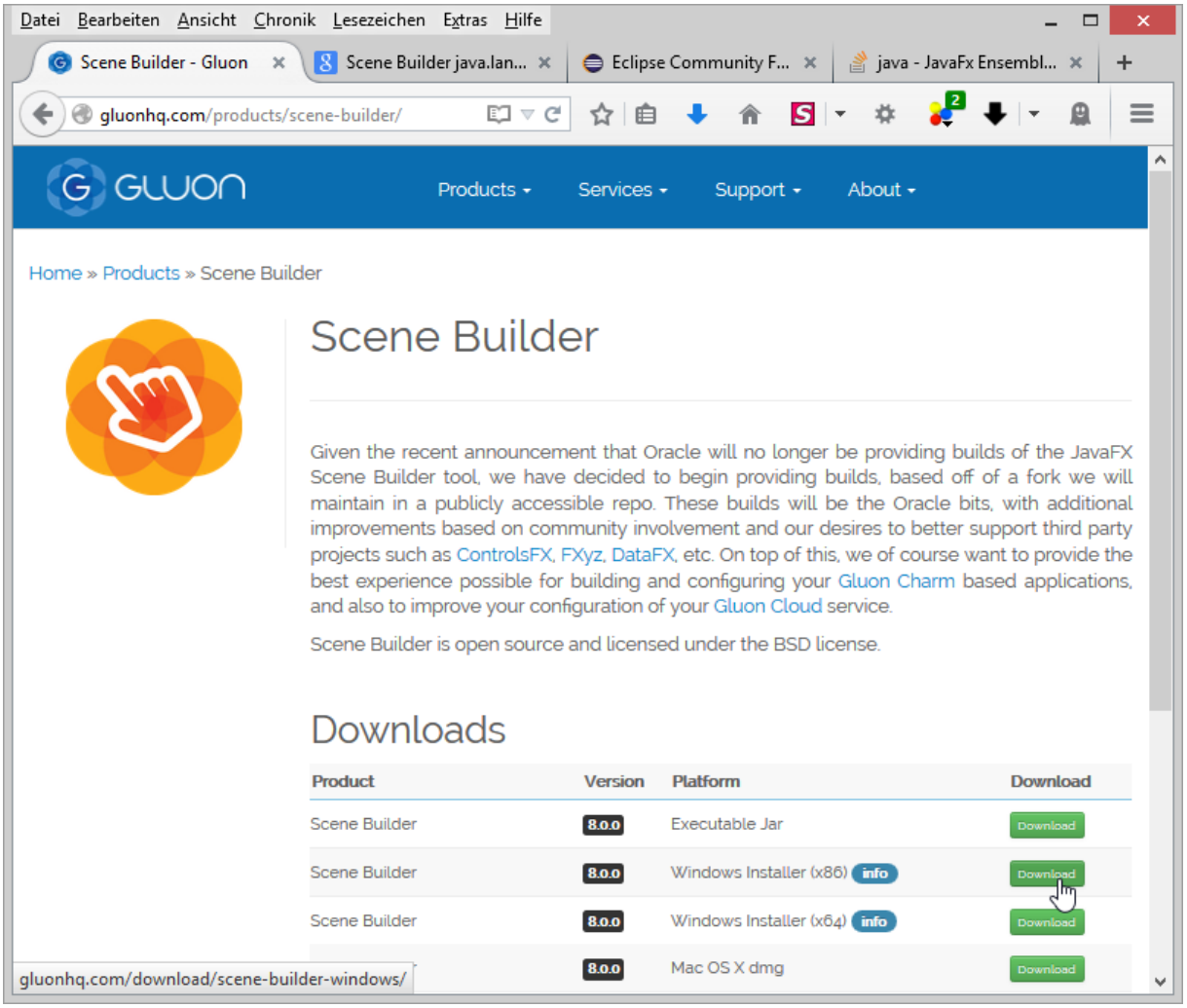
Alternativ befindet sich der Zugang auch auf der Hauptseite unter „Products –> Scene Builder“.



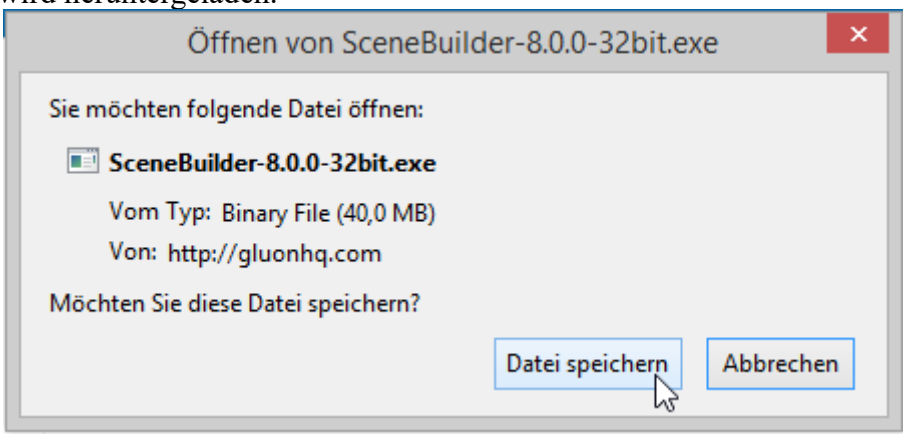
Auf der Web-Seite werden verschiedene Installationsmöglichkeiten für unterschiedlichste Betriebssysteme angeboten. Am flexibelsten ist wahrscheinlich die Nutzung der jar-Datei, die unabhängig vom Betriebssystem verwendbar ist. Diese wird in der konkreten Situation allerdings nicht genutzt, da hierfür keine Integration mit NetBeans existiert.



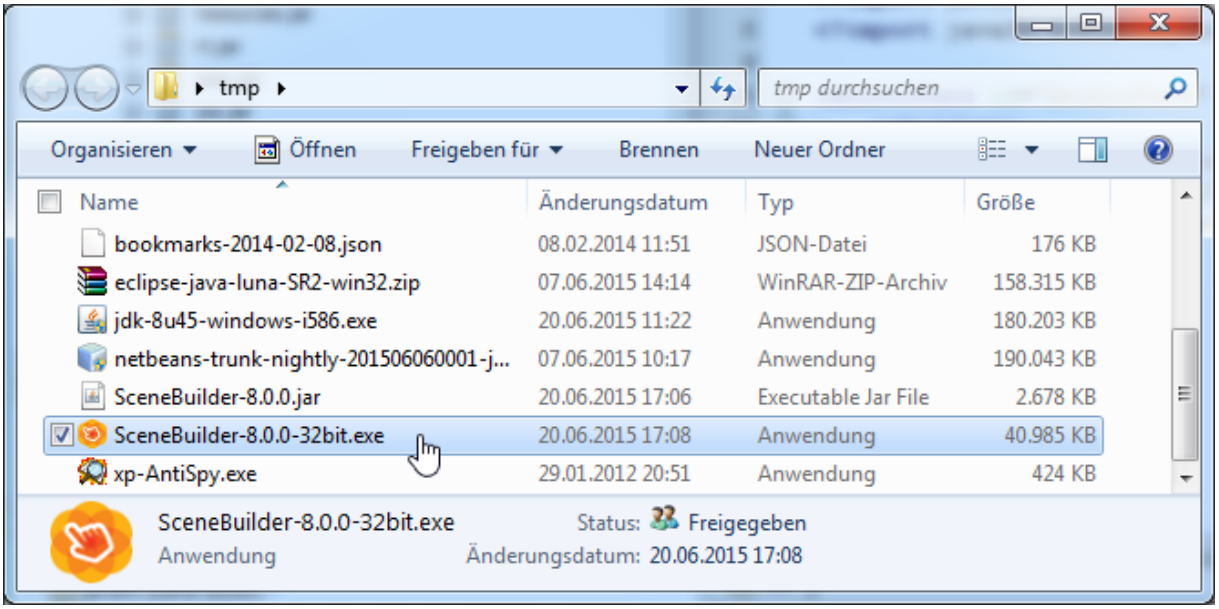
Zur Integration des Scene Builder in NetBeans oder Eclipse wird eine vollständige Installation benötigt, wie sie auch von Gluon angeboten wird. Es muss eine zur aktuellen Java-Version passende Installation gewählt werden, wobei die x86-Variante auch mit einem 64-Bit-Java zusammen laufen sollte.



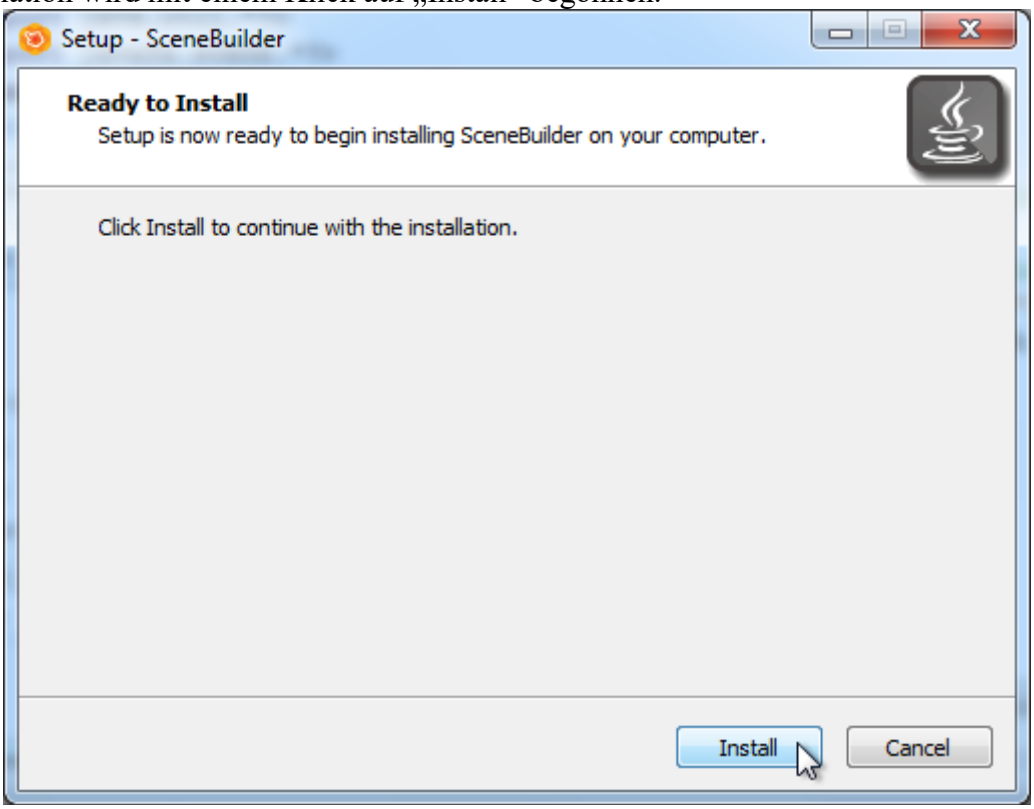
Die exe-Datei wird heruntergeladen.



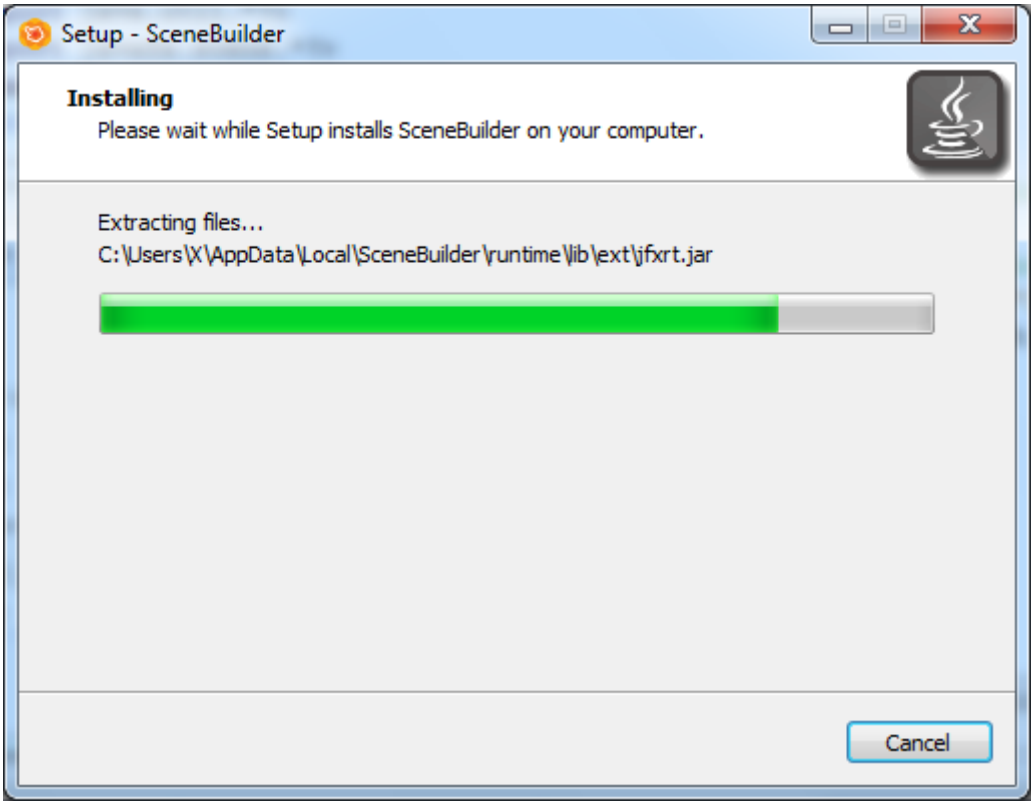
Danach wird die Installation über einen Doppelklick auf der Datei gestartet. Auf etwaige Einstellungen bei der im Rechner genutzten Sicherheitssoftware wird hier nicht eingegangen.



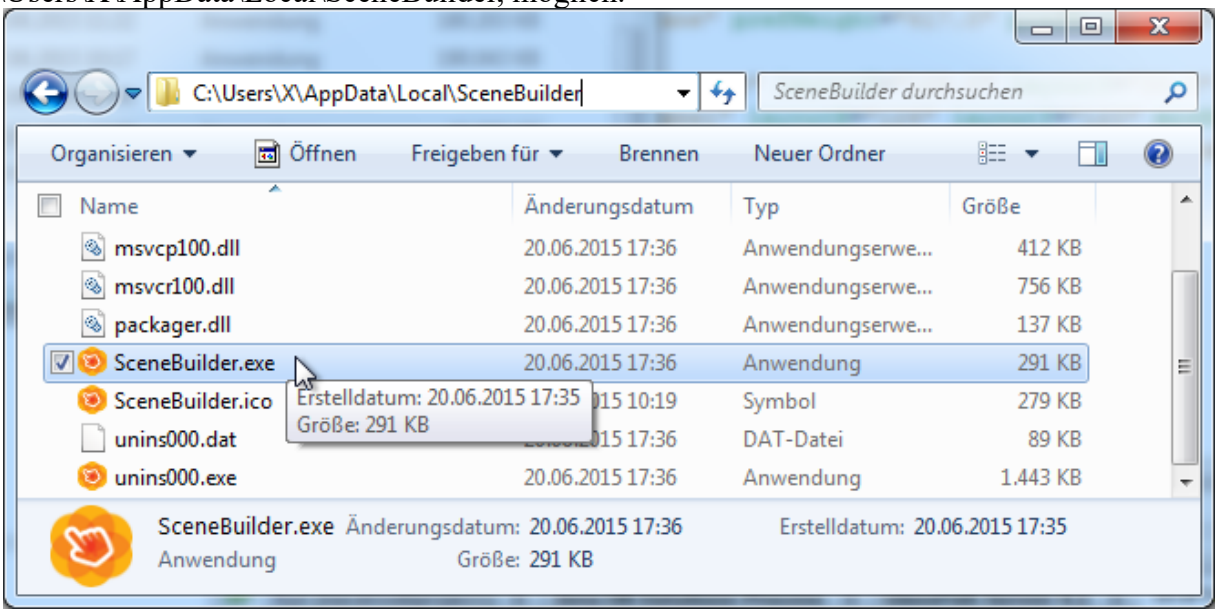
Die Installation wird mit einem Klick auf „Install“ begonnen.



Irritierend ist eine Einstellung des Installationspfades nicht möglich. Man sollte sich den beim Installationsprozess angezeigten Pfad allerdings merken.

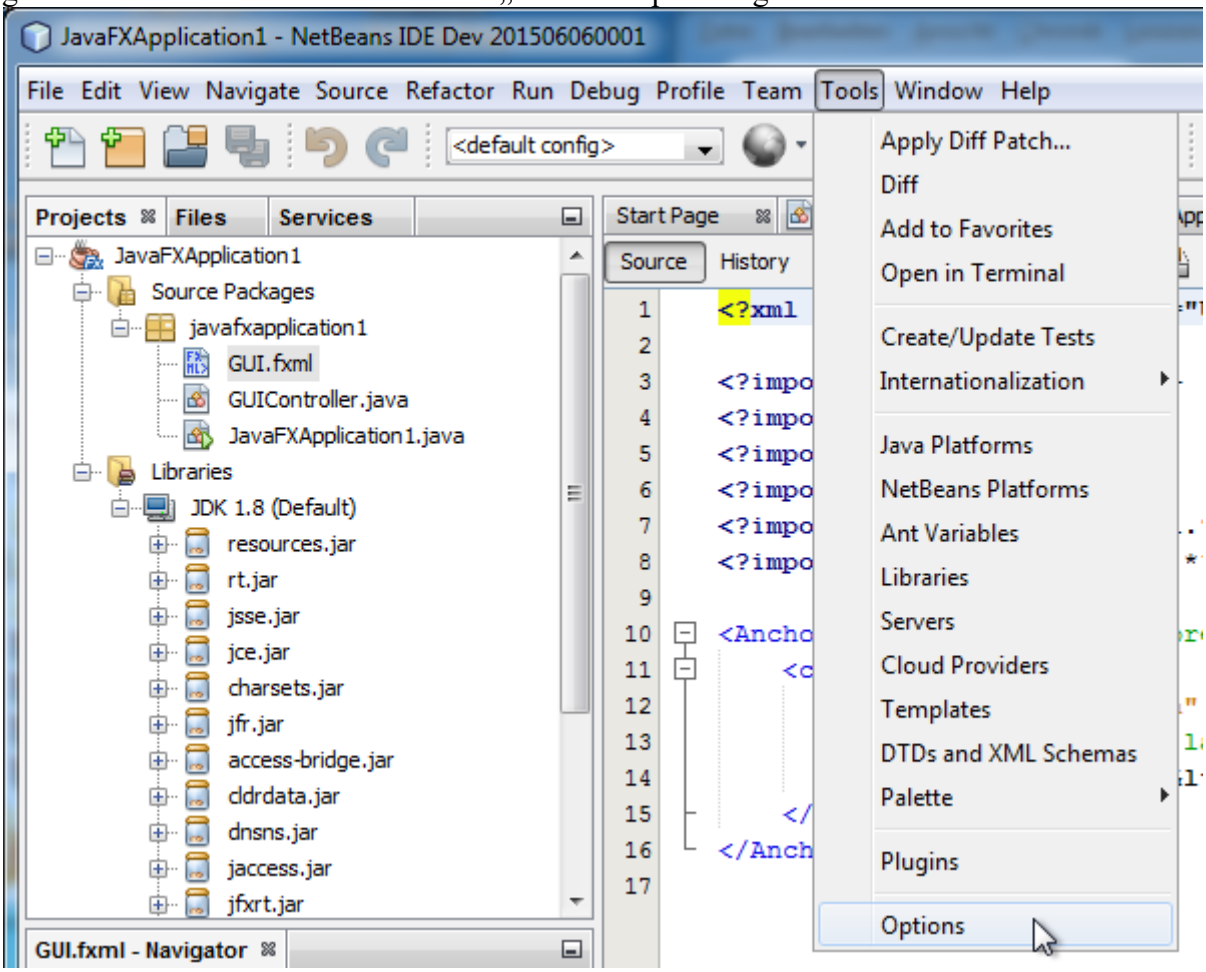


Danach öffnet sich der Scene Builder automatisch.
Ein Start des Scene Builders ist ebenfalls aus seinem Installationsverzeichnis, hier C:\Users\X\AppData\Local\SceneBuilder, möglich.

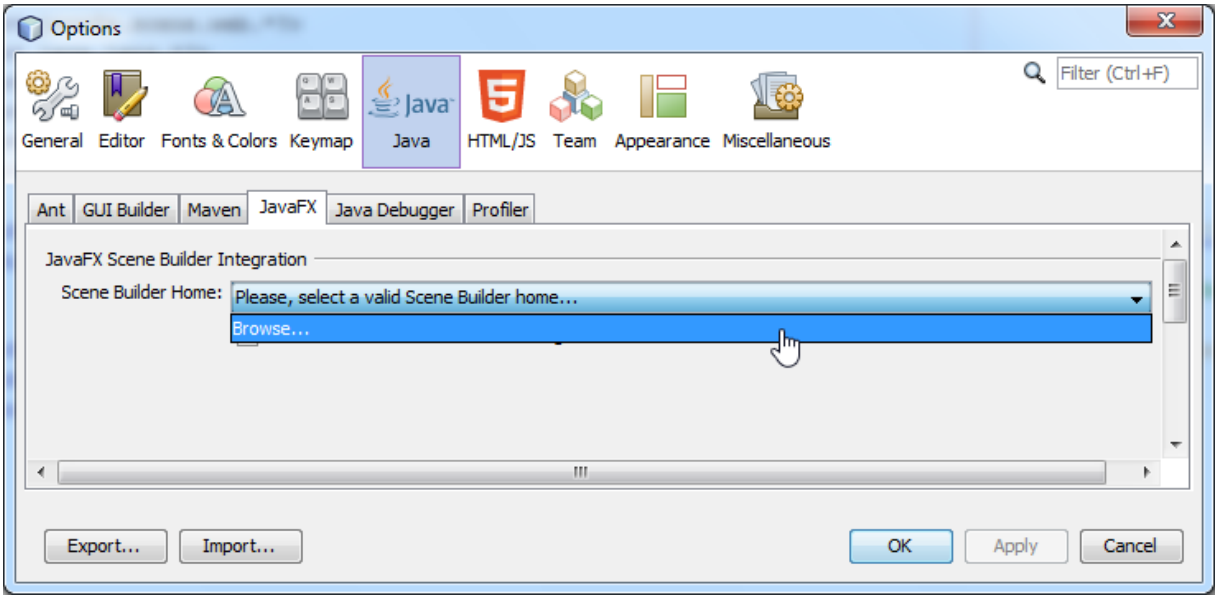


Weiterhin kann das Verzeichnis SceneBuilder nach der Installation in ein anderes Verzeichnis mit Ausführungsberechtigung für den Nutzer verschoben werden. Die Installation nutzt ihre eigene Java-

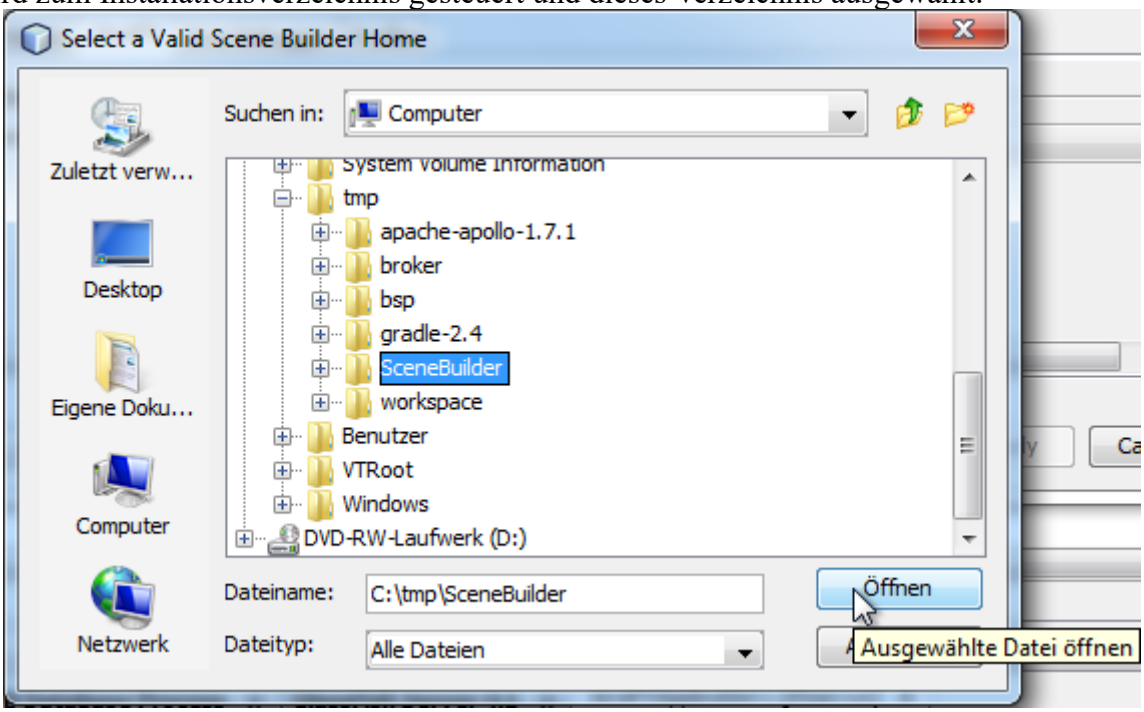
Runtime (JRE), so dass hier keine Versionskonflikte auftreten. Das Verzeichnis wird hier z. B. nach C:\tmp verschoben.
Der Scene Builder muss in NetBeans bekannt gemacht werden, damit er direkt aus NetBeans heraus aufgerufen werden kann. Hierzu wird auf „Tools -> Options“ geklickt.



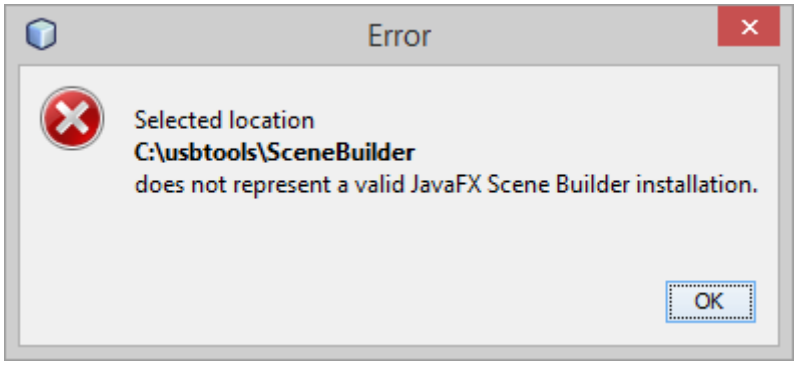
Oben wird zunächst „Java“ sowie der Reiter „JavaFX“ angeklickt und auf den kleinen Pfeil rechts am Rand des Auswahl-Box mit der Beschriftung „Please, select a valid Scene Builder home…“ geklickt und „Browse…“ gewählt.



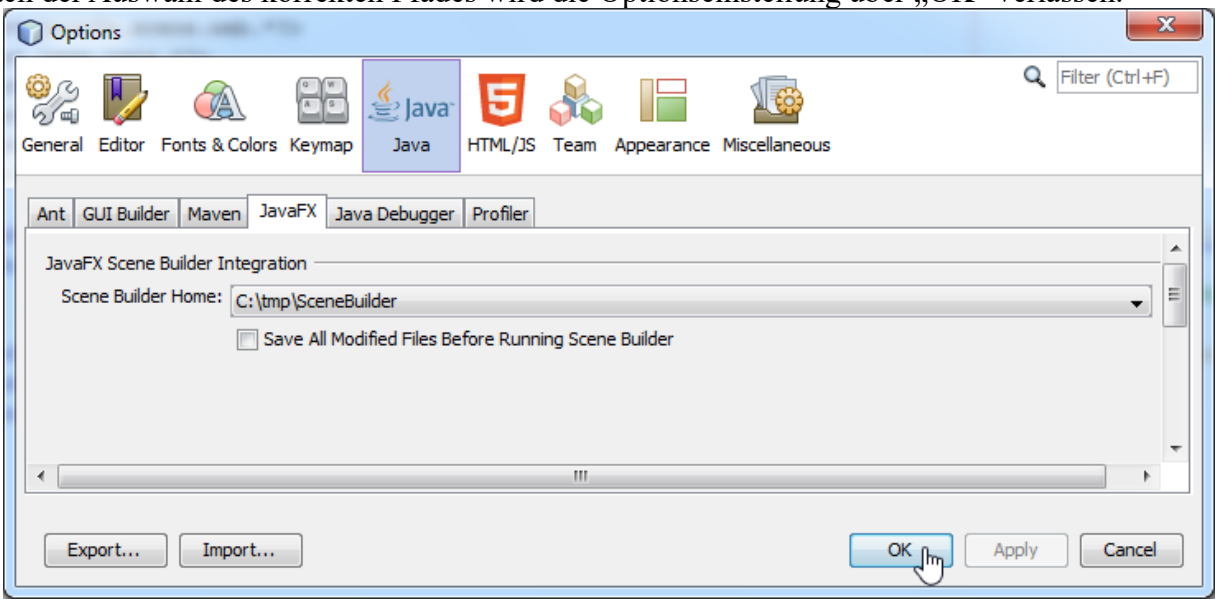
Es wird zum Installationsverzeichnis gesteuert und dieses Verzeichnis ausgewählt.



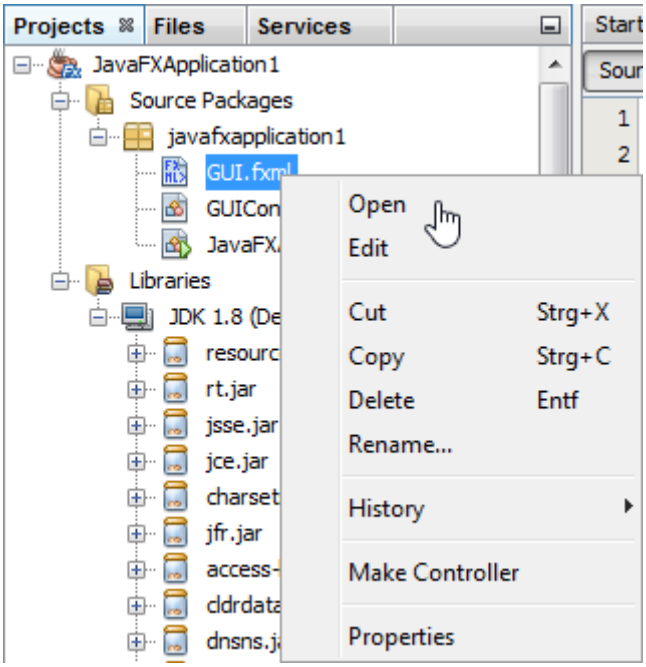
Falls es Probleme mit dem Verzeichnis gibt, wird dieses gemeldet. Eine typische Fehlerquelle ist, dass die in 11.1 beschriebene Aktualisierung nicht stattgefunden hat. Alternativ muss auch eine neuere Java-Version installiert und in NetBeans integriert werden.



Nach der Auswahl des korrekten Pfades wird die Optionseinstellung über „OK“ verlassen.



Wird danach ein Rechtsklick auf einer FXML-Datei gemacht, wird neben „Edit“ auch der Punkt „Open“ angeboten, mit dem dann der Scene Builder aufgerufen wird. Alternativ wird der Builder über einen Doppelklick aufgerufen. Zum Aufruf des Texteditors muss dann immer „Edit“ nach einem Klick mit der rechten Maustaste auf der Datei aufgerufen werden.

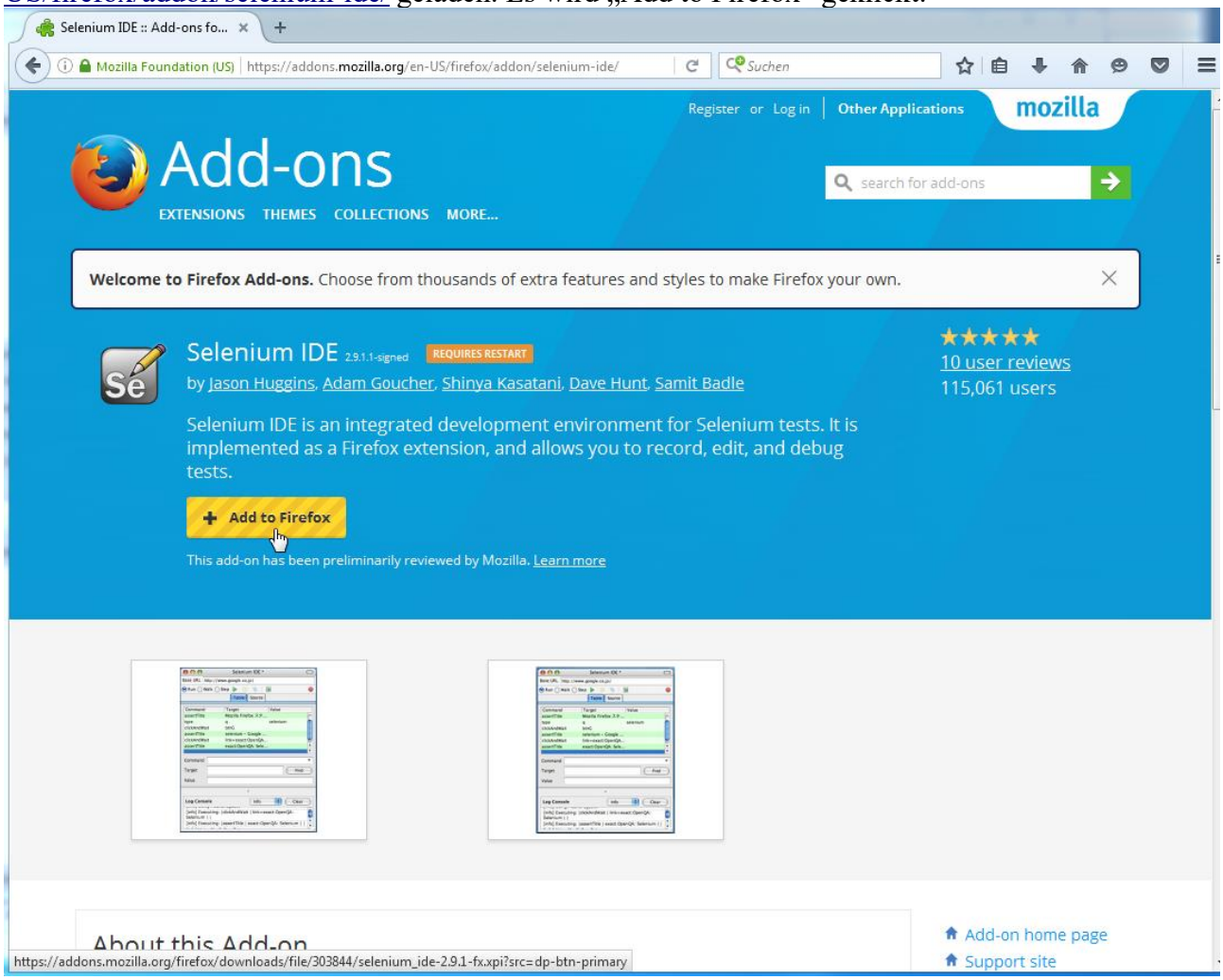

Projects
Files
Services
JavaFXApplication1
Source Packages
javafxapplication1
GUI.fxml
GUICon
JavaFX
Libraries
JDK 1.8 (De
resourc
rt.jar
jsse.jar
jce.jar
charset
jfr.jar
access-
cldrdata
dnsns.j
Open
Edit
Cut Strg+X
Copy Strg+C
Delete Entf
Rename...
History
Make Controller
Properties

12 Selenium

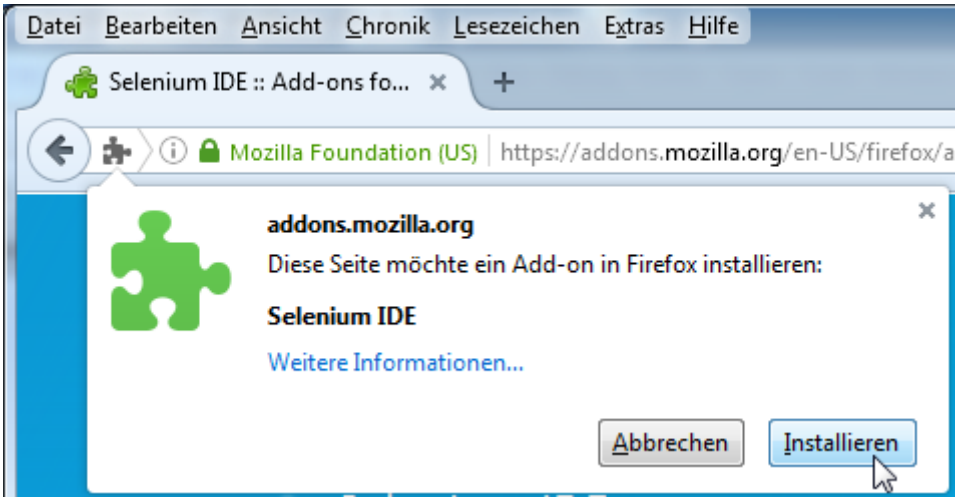
Mit dem Werkzeug Selenium besteht die Möglichkeit, Web-Seiten aus dem Browser heraus zu testen. Grundsätzlich bietet Selenium zwei Werkzeugvarianten an. Mit der Selenium IDE können Nutzeraktionen direkt im Firefox aufgezeichnet und wieder ausgeführt werden. Mit Selenium WebDriver besteht die Möglichkeit, Aktionen für verschiedene Browser zu programmieren und die Ergebnisse zu prüfen. Damit ist Selenium WebDriver sehr gut für professionelle Tests geeignet, IDE eher für kleinere kurzfristige Experimente.

12.1 Selenium IDE

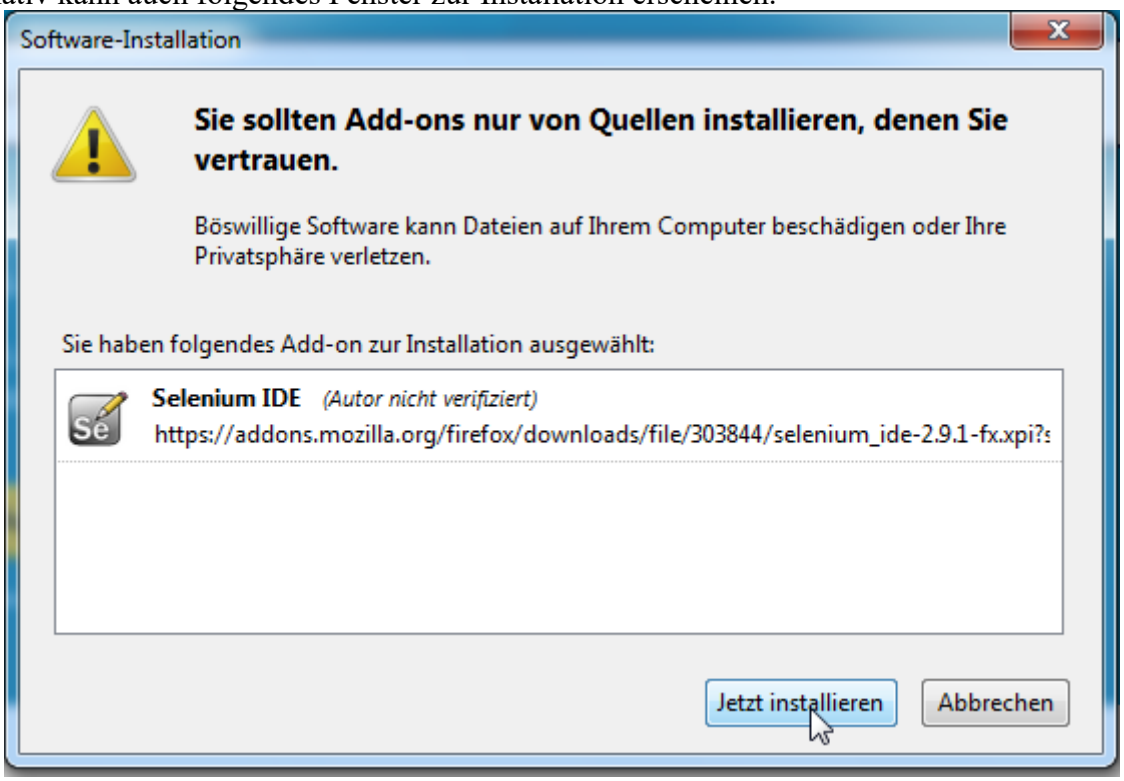
Selenium IDE wird direkt als Firefox-Plugin von https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/ geladen. Es wird „Add to Firefox“ geklickt.



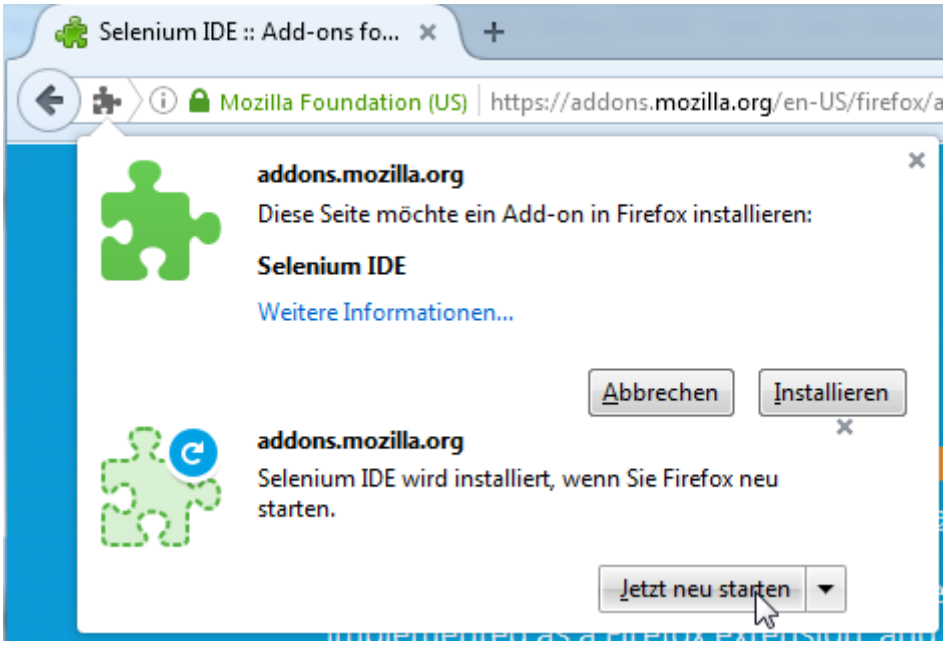
Es wird auf „Installieren“ geklickt.



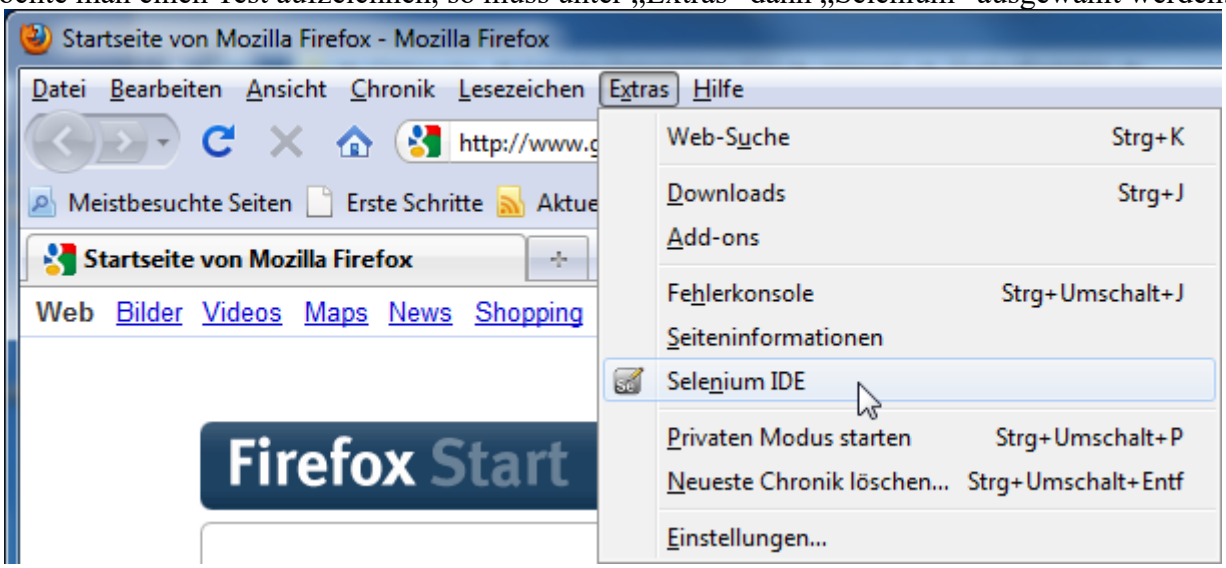
Alternativ kann auch folgendes Fenster zur Installation erscheinen.



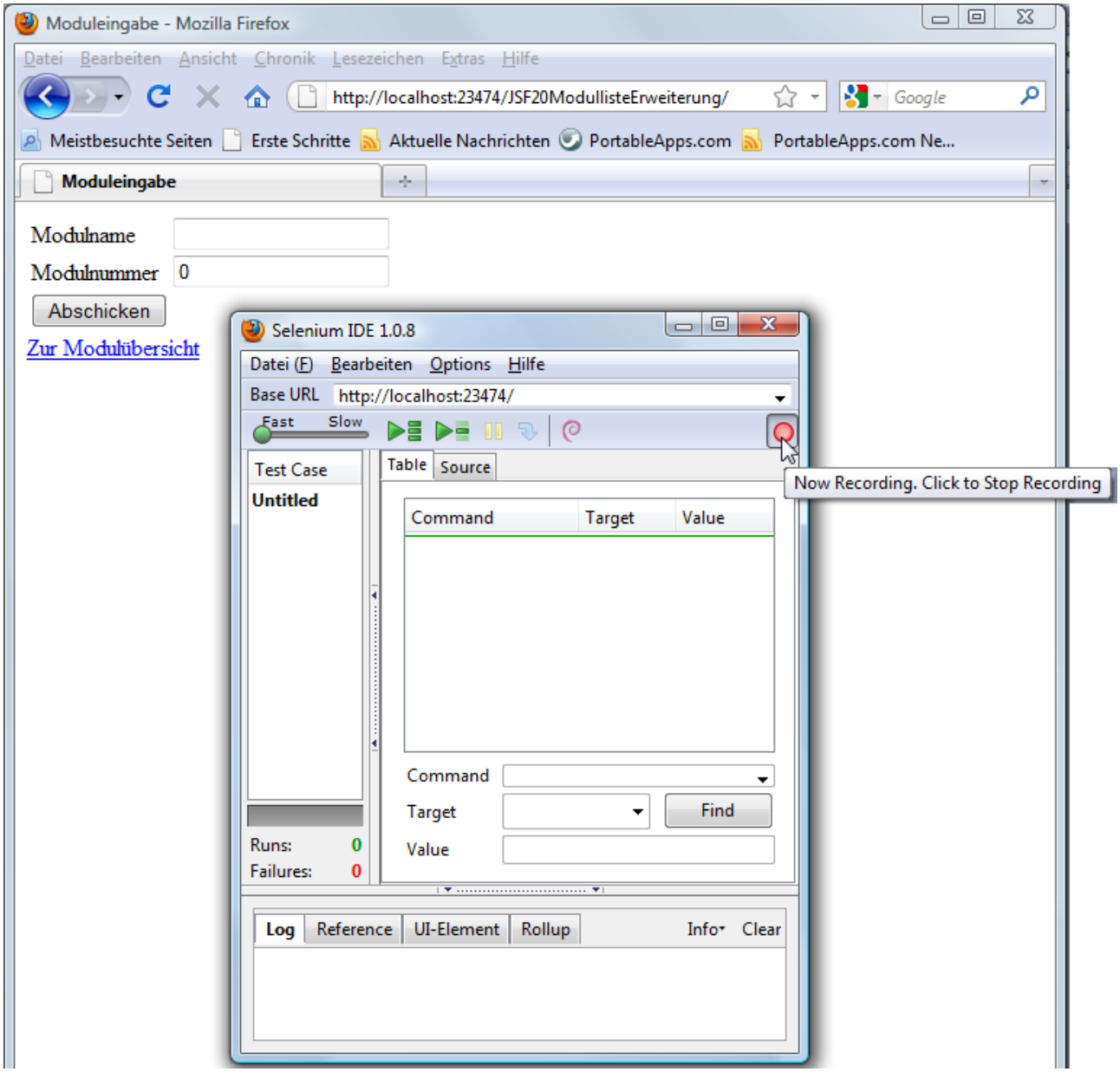
Zur eigentlichen Installation wird Firefox neu gestartet.



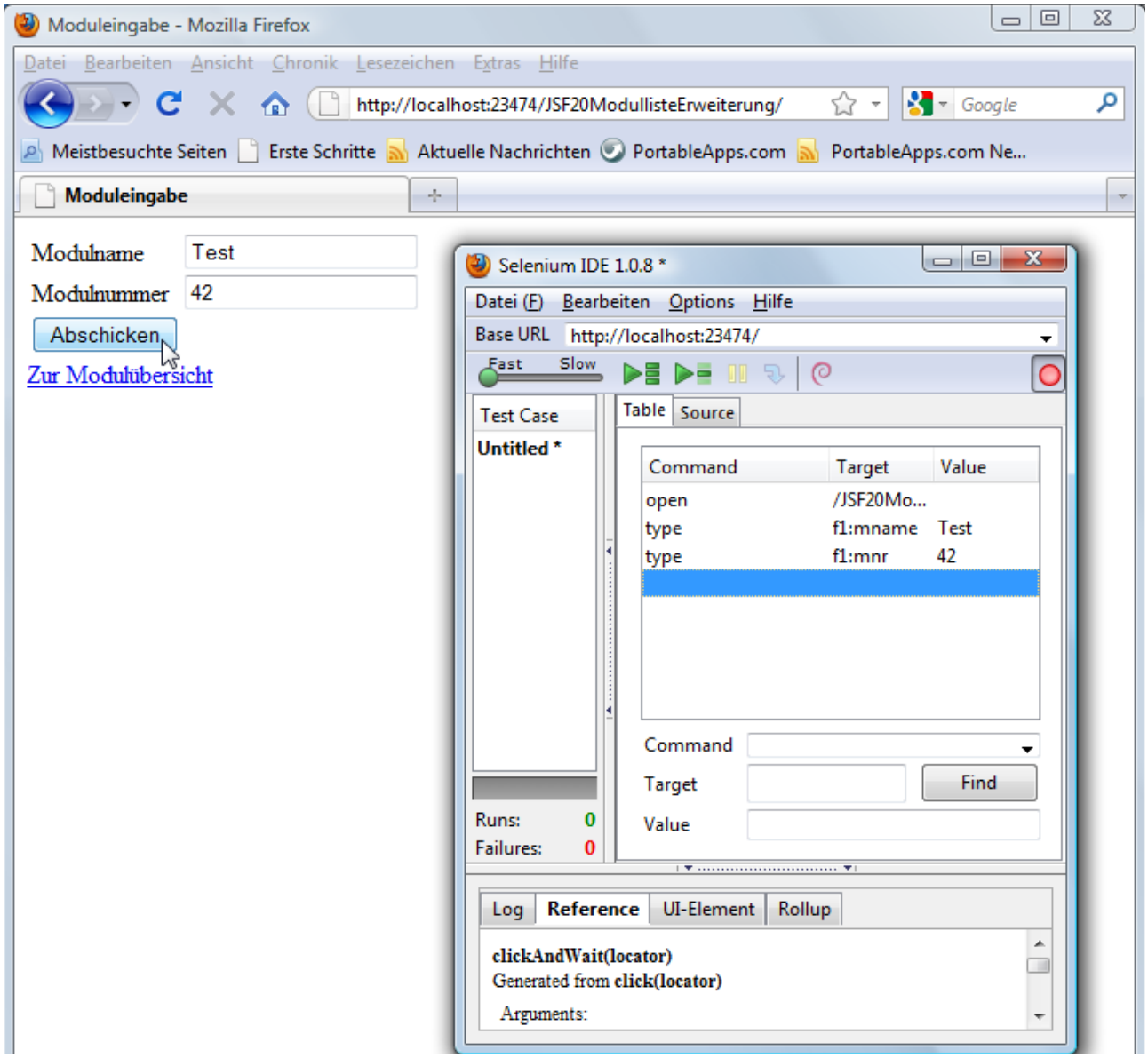
Möchte man einen Test aufzeichnen, so muss unter „Extras“ dann „Selenium“ ausgewählt werden.



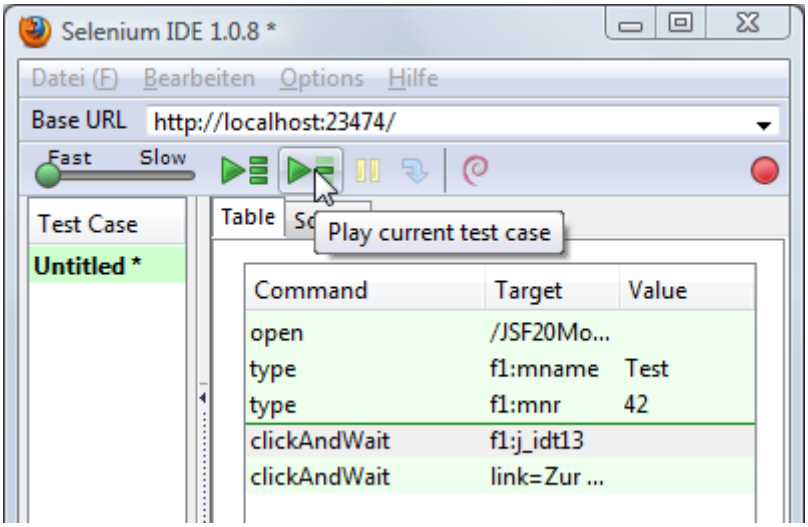
Ab jetzt werden alle Aktionen in Firefox aufgezeichnet, solange bis der Stop/Record-Knopf gedrückt wird.



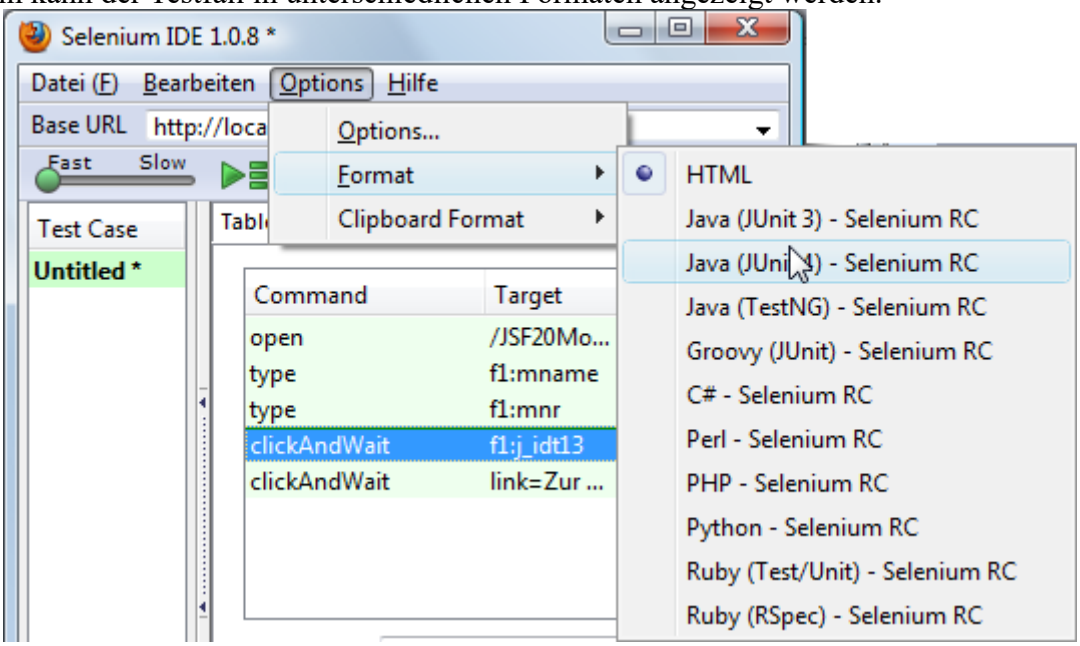
Man erkennt im Selenium-Fenster im Reiter Table die bisher ausgeführten Befehle, im Beispiel die Eingabe der URL, und von „Test sowie „42“.



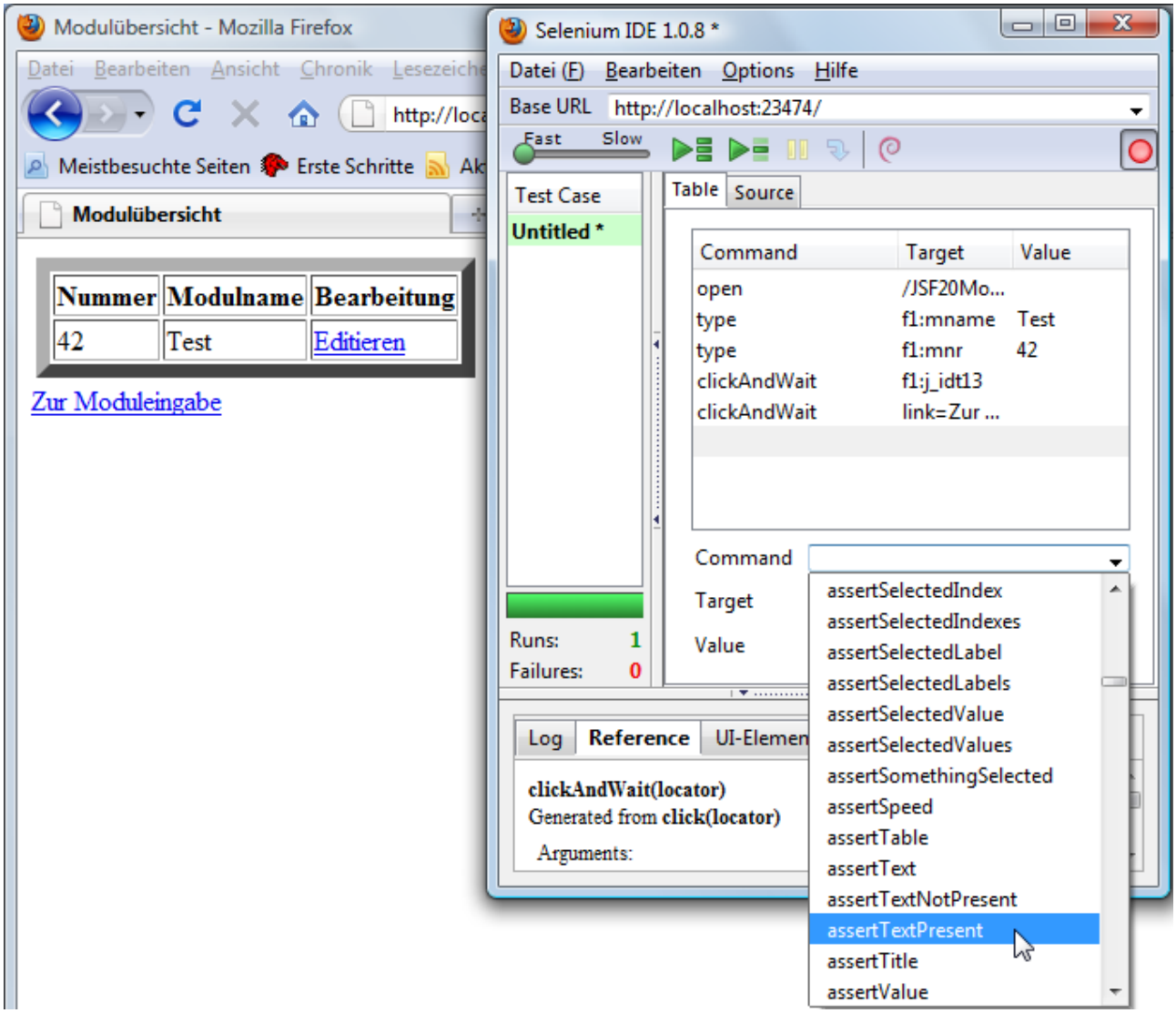
Nach Abschluss der Aufzeichnung kann man sich den Testfall (noch ohne Zusicherungen) einfach abspielen.



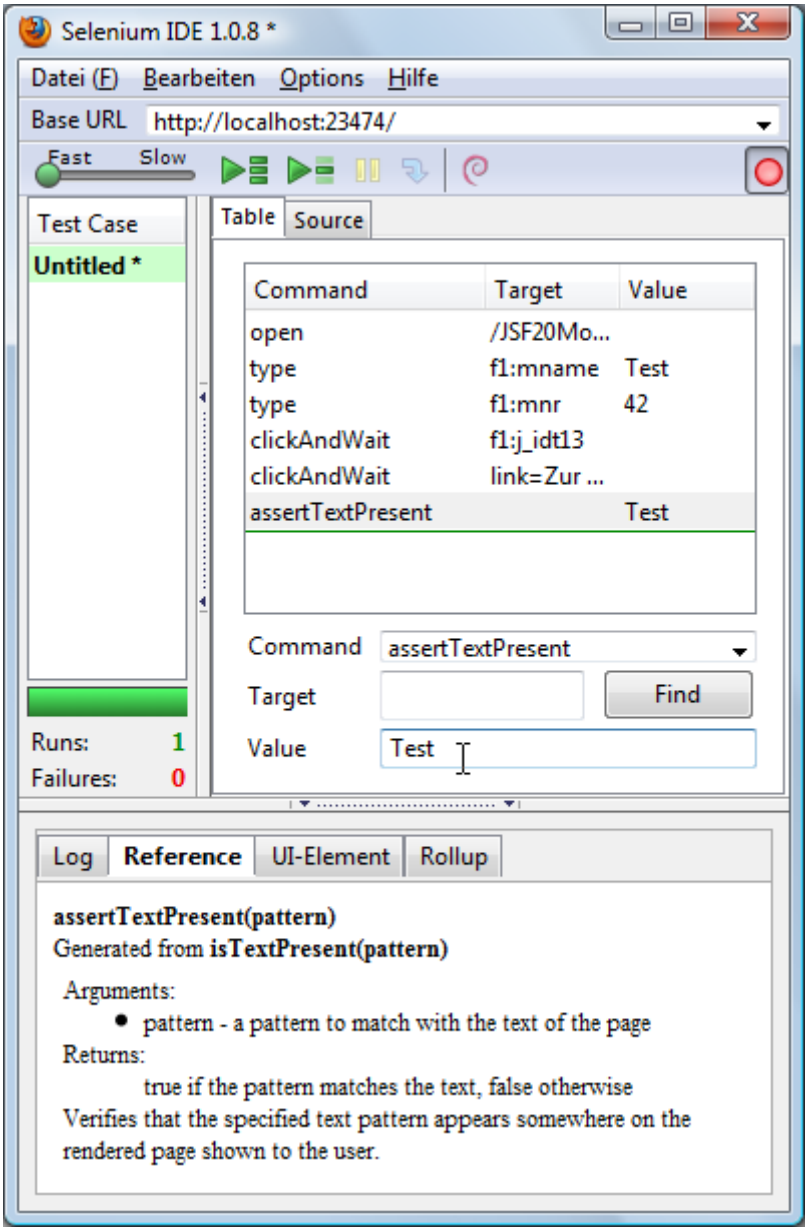
Weiterhin kann der Testfall in unterschiedlichen Formaten angezeigt werden.



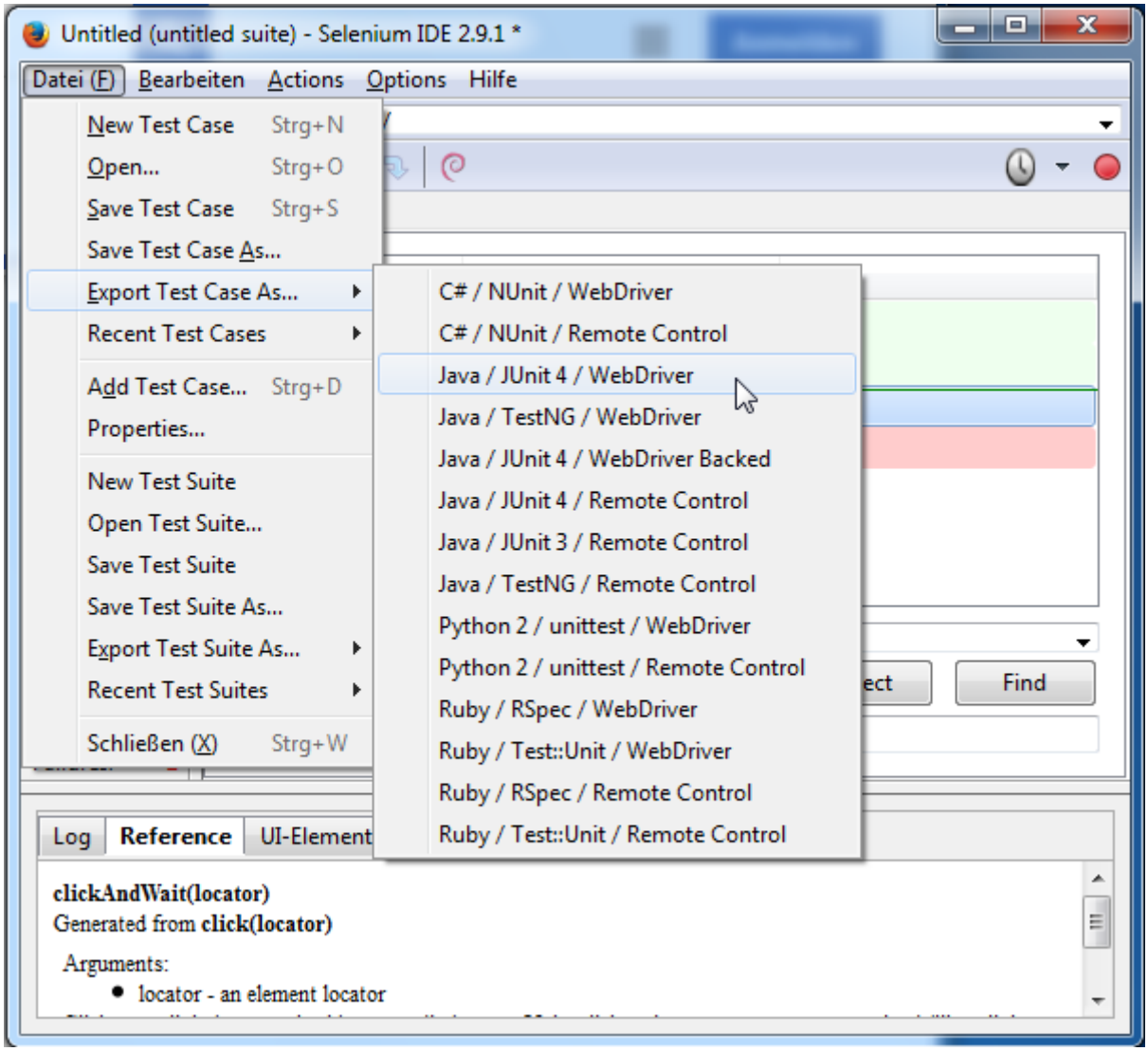
Statt Aktionen aufzuzeichnen, können diese auch manuell ergänzt werden. Eine spezielle Art der Ergänzung ist die Möglichkeit eine Zusicherung hinzuzufügen. Dazu geht man in der üblichen HTML-Tabellenansicht in die relevante Zeile, typischerweise die letzte, und wählt einen Befehl im unteren Menü aus.



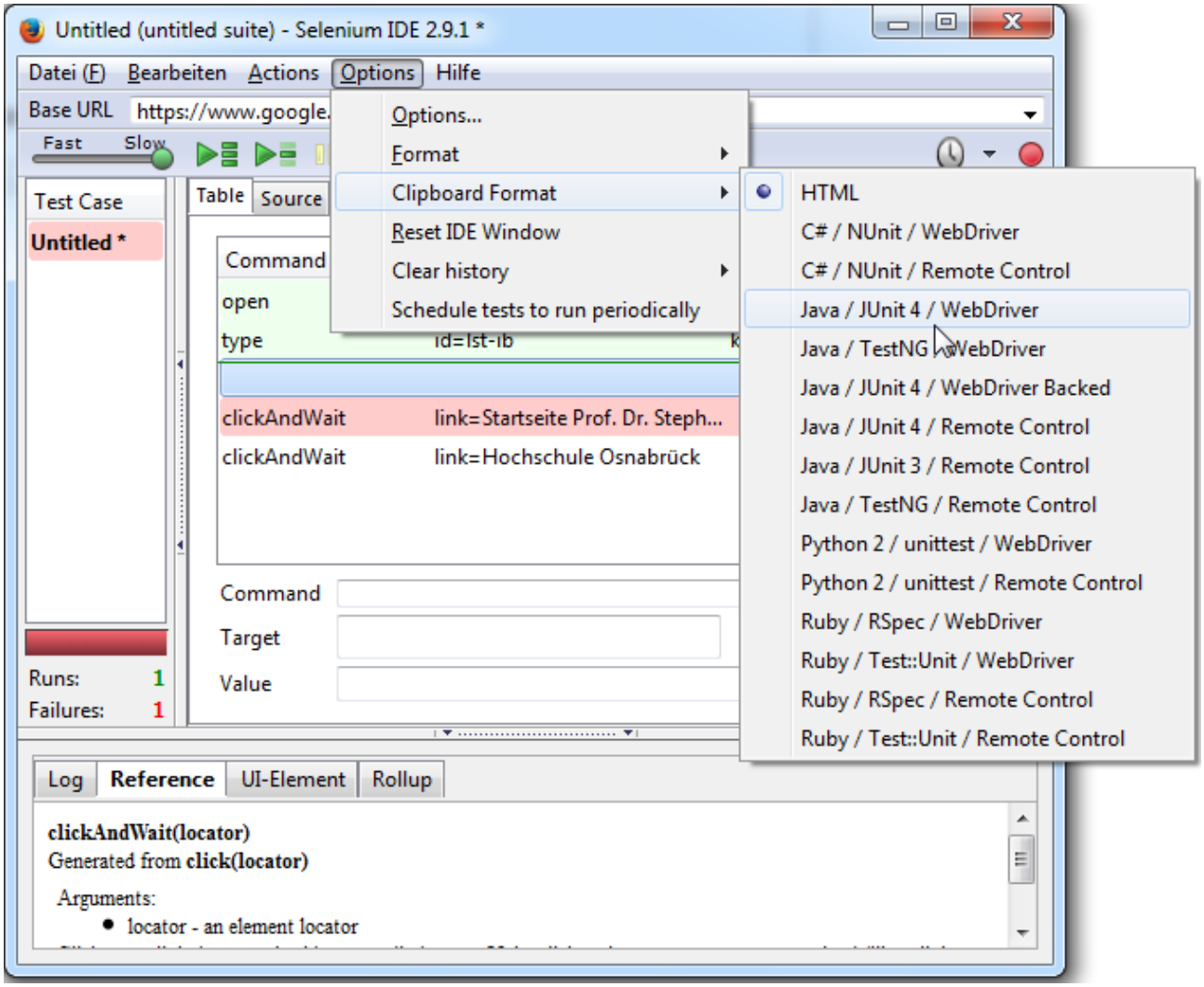
Die Beispielanforderung fordert nur, dass irgendwo der Text „Test“ auf der aktuellen Seite vorkommt. Generell kann man die Testaufzeichnung auch stoppen, um Befehle von Hand einzugeben. Die Aufnahme wird durch erneutes Drücken des roten Knopfes an der Stelle fortgesetzt, an der der Cursor in der Tabelle steht. Man beachte, dass im unteren Reference eine kurze Information zur Syntax und Semantik des Befehls steht.



Abgeschlossene Testfälle können vielfältig verwaltet werden. Besonders interessant ist die Möglichkeit den Test nach JUnit oder TestNG zu exportieren. Dabei kann entweder der Export in Selenium WebDriver, siehe nächstes Unterkapitel oder Remote Control, um den aufgezeichneten Test direkt abspielen zu können, interessant sein.



Alternativ kann man ggfls. auch Testfälle ins Clipboard, die Zwischenablage, kopieren und dann in JUnit nutzen.

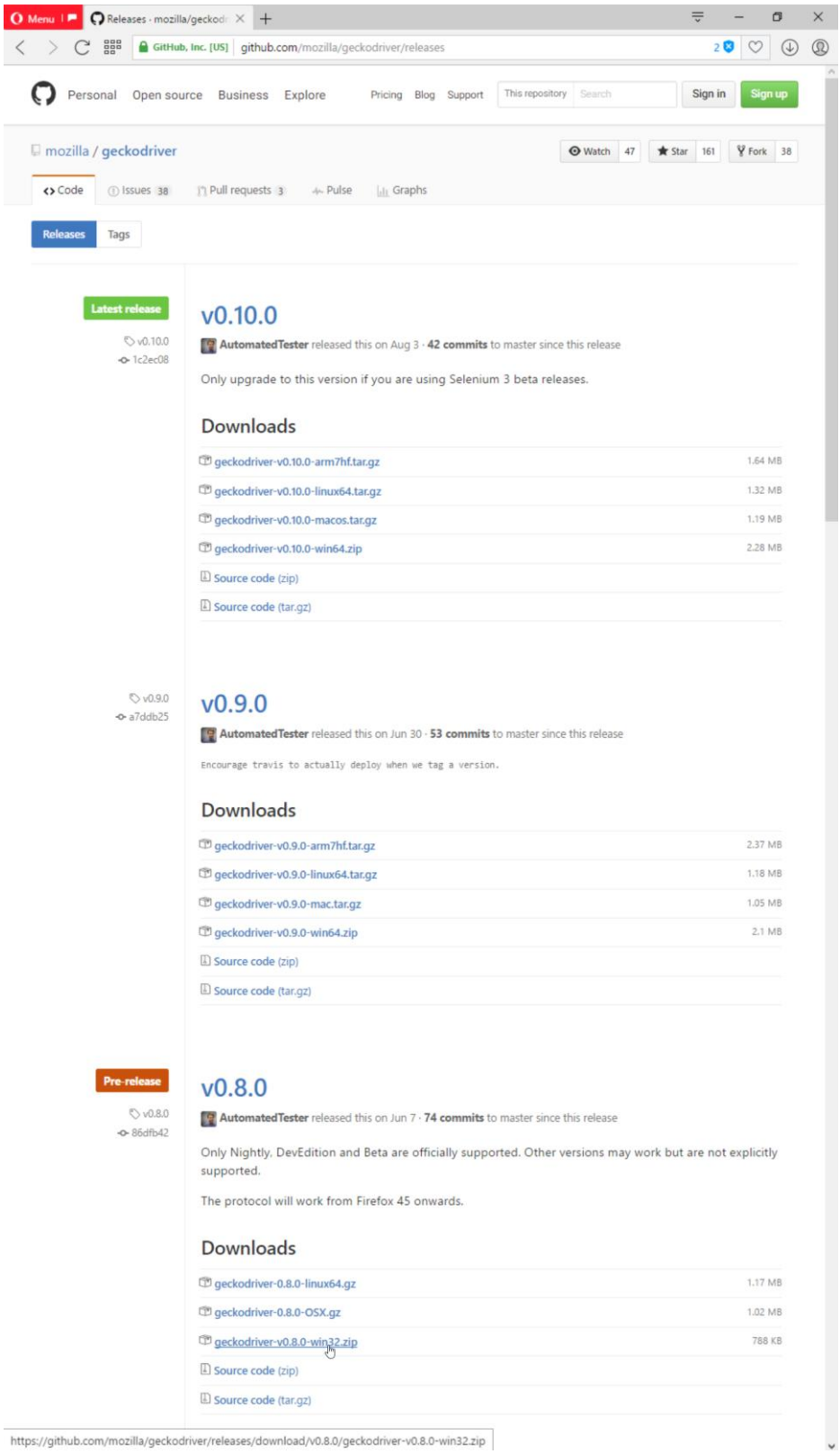


12.2 Selenium WebDriver

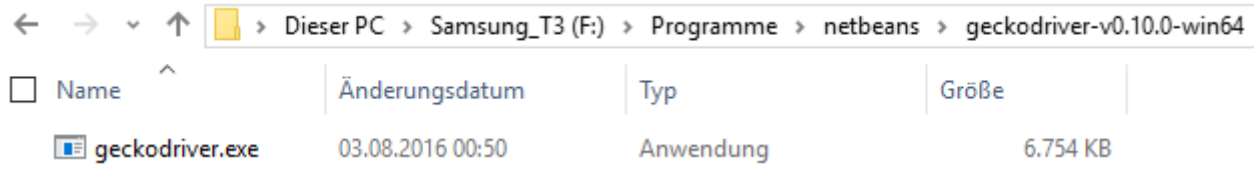
Mit Selenium WebDriver können Tests programmiert werden, mit denen ein Browser gesteuert werden kann. Dabei gibt es eine größere Anzahl nutzbarer Browser, deren detailliertes Verhalten von der genutzten Web-Engine abhängt. Weiterhin unterstützen Browser Nutzerprofile mit individuellen Einstellungen, die sich auf die Nutzung des Werkzeugs auswirken können. Die allgemeine Empfehlung ist deshalb, einen Browser speziell zum Testen zu nutzen, was z. B. in einer virtuellen Maschine möglich ist.

Ab der Firefox-Version 48 wird ein zusätzliches Programm zur Nutzung von NetBeans namens Marionette benötigt, das von der Webseite https://github.com/mozilla/geckodriver/releases geladen werden kann. Es ist zu beachten die passende Version zum Betriebssystem, auch 64- oder 32-Bit zu laden. Eventuell stehen nur ältere Versionen für 32-Bit (x86) zur Verfügung. Da zum Zeitpunkt der Erstellung dieses Dokuments gerade Selenium 3 eingeführt wurde, sind einige Änderungen und Korrekturen zu erwarten. Für eine professionelle Nutzung von NetBeans sollten nur Browser-

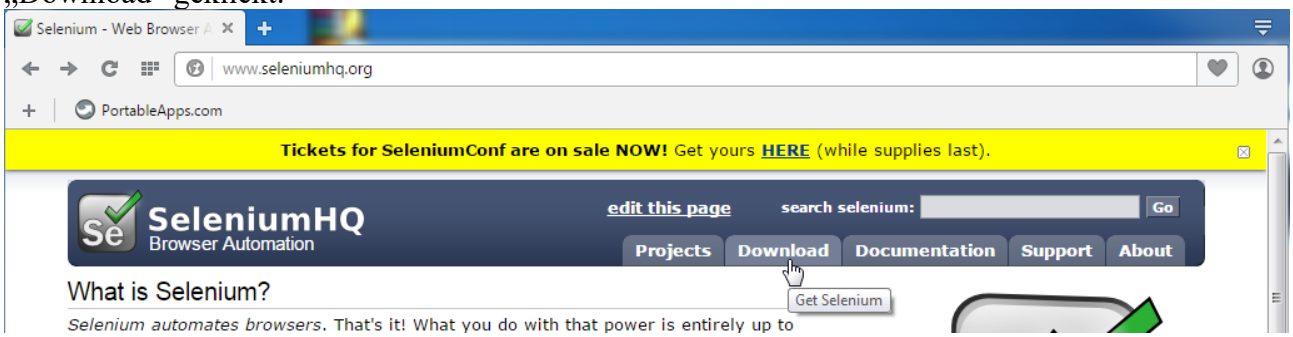
Versionen genutzt werden, die vor der Veröffentlichung der genutzten Selenium-Version entstanden sind.



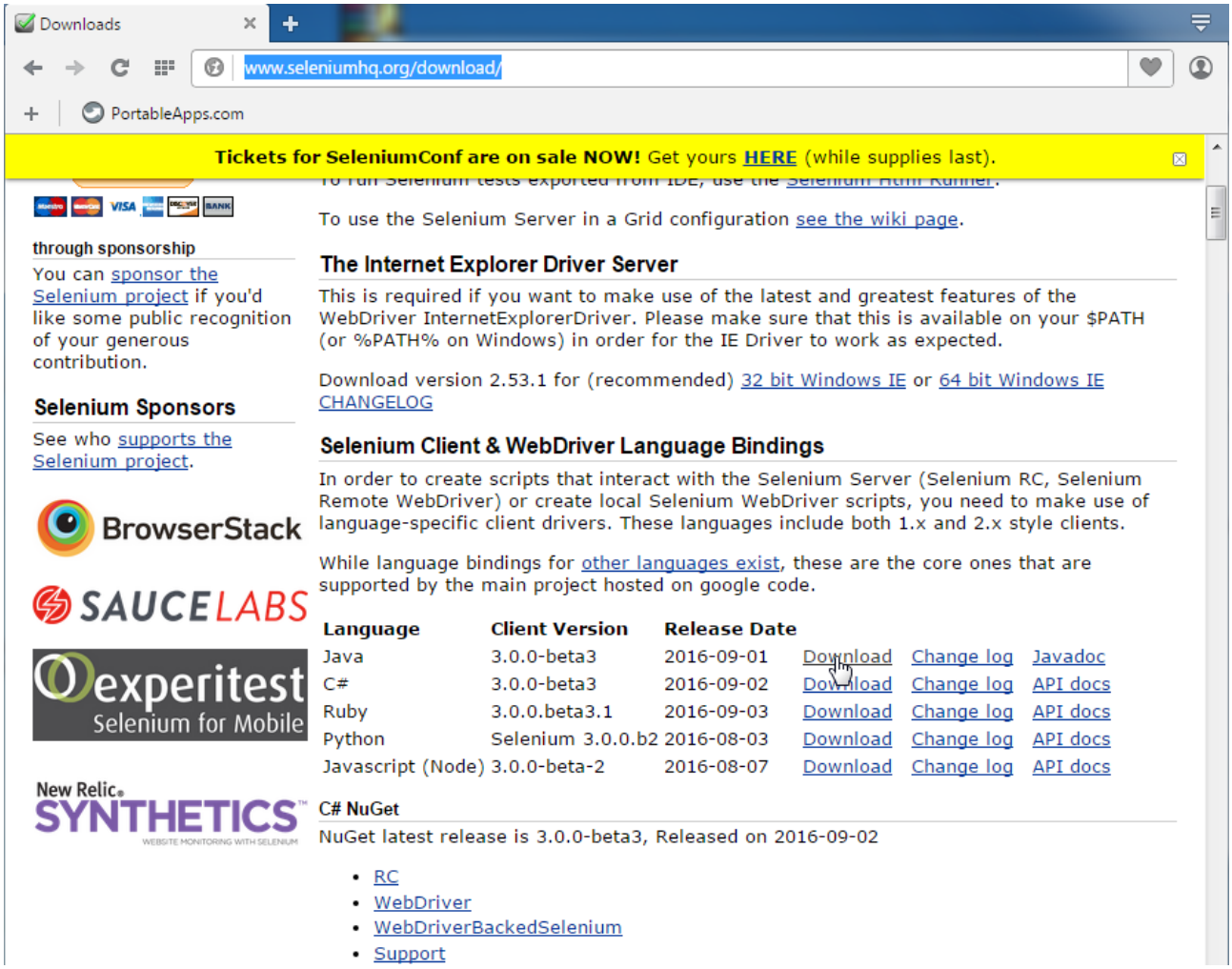
Die erhaltene Zip-Datei wird an einem ausführbaren Ort ausgepackt.



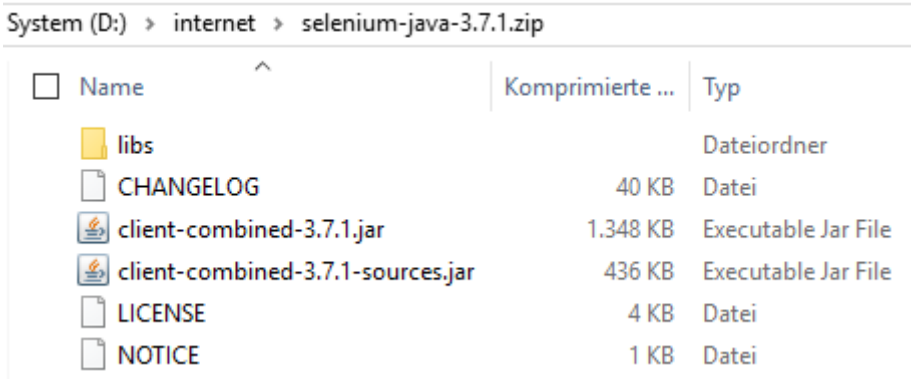
Zum Download von WebDriver wird auf der Selenium-Seite http://www.seleniumhq.org wieder auf „Download“ geklickt.



Auf der Seite wird nach „Selenium Client & WebDriver Language Bindings gesucht und bei der passenden Version auf „Download“ geklickt. Es ist sinnvoll die Seite http://www.seleniumhq.org/download/ intensiver zu studieren, da hier für weitere Browser benötigte WebDriver angeboten werden, die zusätzlich zu installieren sind. Für den Internet Explorer wird weiterhin ein zusätzlicher Server-Prozess benötigt.

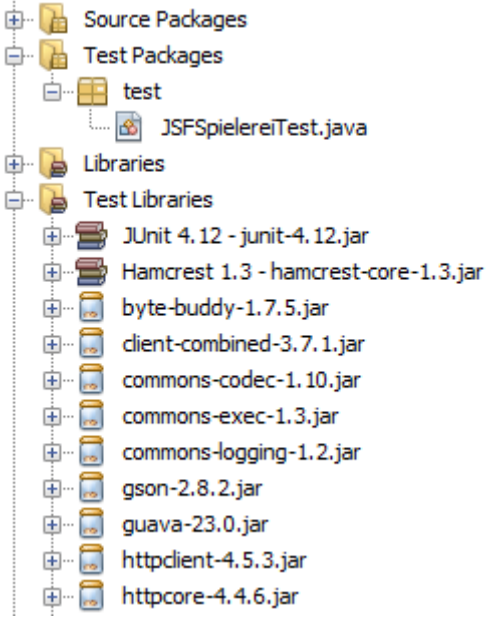


Die heruntergeladene Zip-Datei enthält einige Jar-Dateien in zwei Ordnern die in ein Projekt eingebunden werden müssen, wie es in „4.5 Organisation von Imports“ beschrieben ist.



Name	Änderungsdatum	Typ	Größe
byte-buddy-1.7.5.jar	01.02.1985 00:00	Executable Jar File	2.793 KB
client-combined-3.7.1.jar	01.02.1985 00:00	Executable Jar File	1.431 KB
commons-codec-1.10.jar	01.02.1985 00:00	Executable Jar File	278 KB
commons-exec-1.3.jar	01.02.1985 00:00	Executable Jar File	54 KB
commons-logging-1.2.jar	01.02.1985 00:00	Executable Jar File	61 KB
gson-2.8.2.jar	01.02.1985 00:00	Executable Jar File	228 KB
guava-23.0.jar	01.02.1985 00:00	Executable Jar File	2.554 KB
httpclient-4.5.3.jar	01.02.1985 00:00	Executable Jar File	731 KB
httpcore-4.4.6.jar	01.02.1985 00:00	Executable Jar File	317 KB

Da es im Beispiel nur um Tests geht, werden diese unter „Test Libraries“ eingebunden. Werden alle Jar-Dateien genutzt, ist JUnit automatisch in einer passenden Version enthalten. Die Tests werden in einer Klasse JSFSpielereiTest stehen, was zu folgender Projektstruktur führt. In aktuelleren Versionen können weitere jar-Dateien aus dem lib-Verzeichnis von Selenium hinzukommen.



Die eigentlichen Tests können wie folgt aussehen und werden wie Tests auch und in „5 Nutzung von JUnit (NetBeans 8)“ beschrieben ausgeführt.

```
package test;

import java.util.Date;
import org.junit.After;
import org.junit.Assert;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
```

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.WebDriverWait;

public class JSFSpielereiTest {

    private WebDriver driver;

    @BeforeClass
    public static void setUpOnce() {
        // Pfad an eigene Einstellungen anpassen
        System.setProperty("webdriver.gecko.driver"
                , "F:\\Programme\\netbeans\\geckodriver-v0.10.0-win64\\geckodriver.exe");
    }

    @Before
    public void setUp() {
        DesiredCapabilities capabilities = DesiredCapabilities.firefox();
        capabilities.setCapability("marionette", true);
        driver = new FirefoxDriver(capabilities);
        //driver = new MarionetteDriver(); // veraltet
    }

    @After
    public void tearDown() {
        driver.quit(); // gehoert hier in, fuehrt eventuell zur Exception evtl. @AfterClass
    }

    private void starten() {
        driver.get("http://localhost:8080/JSFSpielerei");
    }

    private void modulEinfuegen(String name, int id) {
        starten();
        WebElement element
                = driver.findElement(By.name("index:mname"));
        element.clear();
        element.sendKeys(name);
        element = driver.findElement(By.name("index:mnr"));
        element.clear();
        element.sendKeys("" + id);
        driver.findElement(By.id("index:abschicken"))
                .click();
    }
```

```
private void warteAufSeiteMitId(String id) {
    (new WebDriverWait(driver, 10)).until(
            new ExpectedCondition<WebElement>() {
        @Override
        public WebElement apply(WebDriver d) {
            return d.findElement(By.id(id));
        }
    });
}

private void ausgabe() {
    warteAufSeiteMitId("index");
    driver.findElement(By.id("index:ausgabelink")).click();
}

private void zurueck() {
    warteAufSeiteMitId("ausgabe");
    driver.findElement(By.id("ausgabe:zurueck")).click();
}

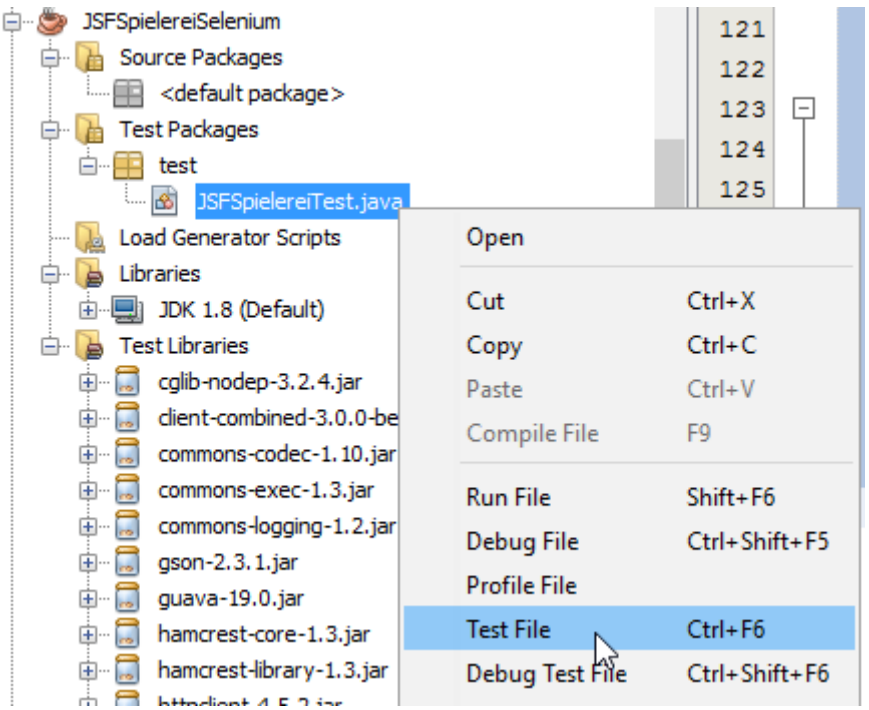
private boolean inTabelle(String name, int id) {
    warteAufSeiteMitId("ausgabe");
    int sizeRows = driver
            .findElements(By
                    .xpath("//table[@id='ausgabe:tabelle']/tbody/tr"))
            .size();
    //System.out.println("size: " + sizeRows);
    for (int tableRow = 1; tableRow <= sizeRows; tableRow++) {
        String rowLocator = String
                .format("//table[@id='ausgabe:tabelle']/tbody/tr[%d]", tableRow);
        String column1 = driver
                .findElement(By.xpath(rowLocator + "/td[1]"))
                .getText()
                .trim();
        String column2 = driver
                .findElement(By.xpath(rowLocator + "/td[2]"))
                .getText()
                .trim();
        //System.out.println(column1 + " " +column2);
        if (name.equals(column2) && ("" + id).equals(column1)) {
            return true;
        }
    }
    return false;
}

@Test
public void erfolgreicheEinUndAusgabeTest() {
    String name = new Date().toString();
```

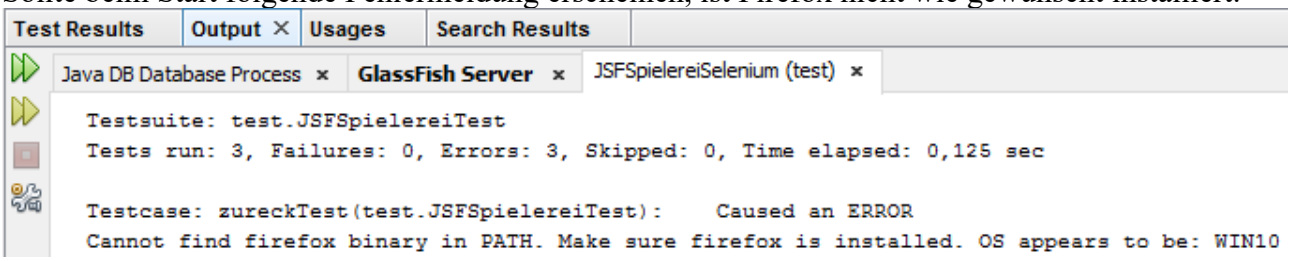
```
        int id = (int) (Math.random() * 100000);
        modulEinfuegen(name, id);
        Assert.assertTrue("Eintrag (" + name + "," + id + ") nicht erfolgt"
                , inTabelle(name, id));
    }

    @Test
    public void ausgabeNutzenTest() {
        String name = new Date().toString();
        int id = (int) (Math.random() * 100000);
        modulEinfuegen(name, id);
        starten();
        ausgabe();
        Assert.assertTrue("Eintrag (" + name + "," + id + ") nicht erfolgt"
                , inTabelle(name, id));
    }

    @Test
    public void zureckTest() {
        String name = new Date().toString();
        int id = (int) (Math.random() * 100000);
        modulEinfuegen(name, id);
        zurueck();
        ausgabe();
        Assert.assertTrue("Eintrag (" + name + "," + id + ") nicht erfolgt"
                , inTabelle(name, id));
    }
}
```

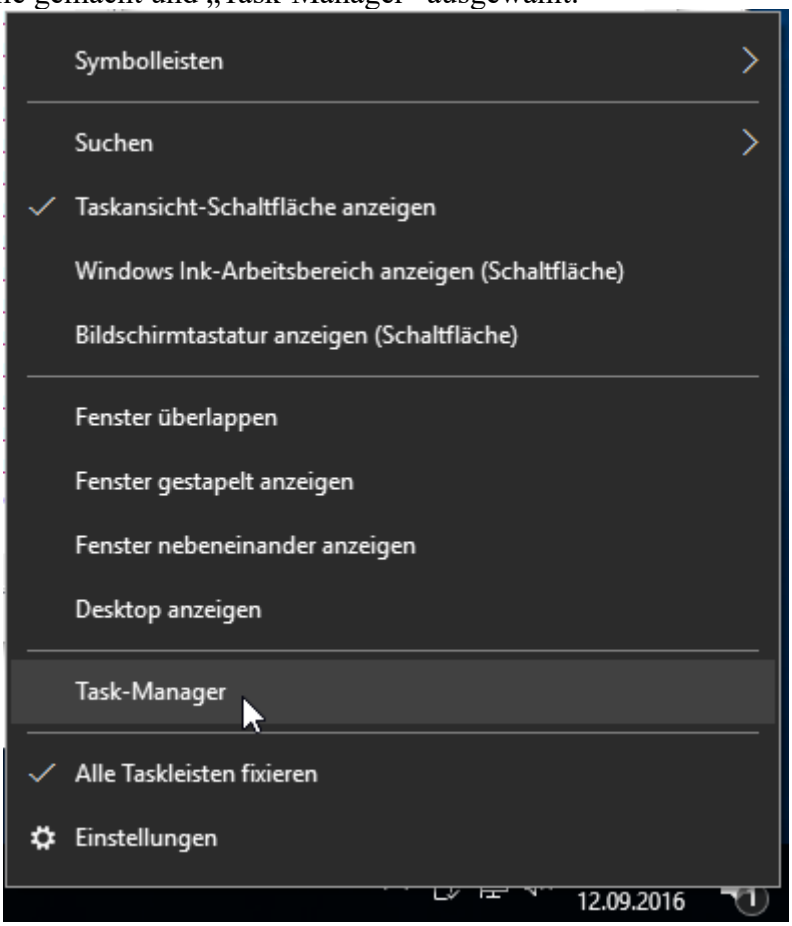


Sollte beim Start folgende Fehlermeldung erscheinen, ist Firefox nicht wie gewünscht installiert.

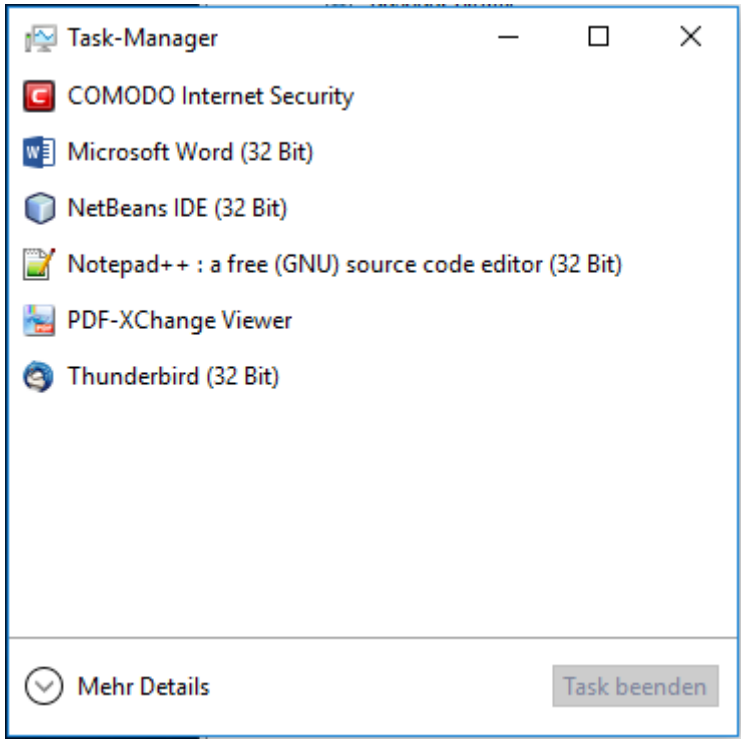


Sollte Firefox ordnungsgemäß installiert sein, ist eine Problemquelle, dass die 64-Bit-Version von Firefox nicht unterstützt wird. Hier ist die einfachste Lösung die 32-Bit-Version zu installieren. Alternative Ansätze z. B. mit portablen Versionen hatten versionsabhängig bisher Probleme, werden aber nach der folgenden Anmerkung vorgestellt.

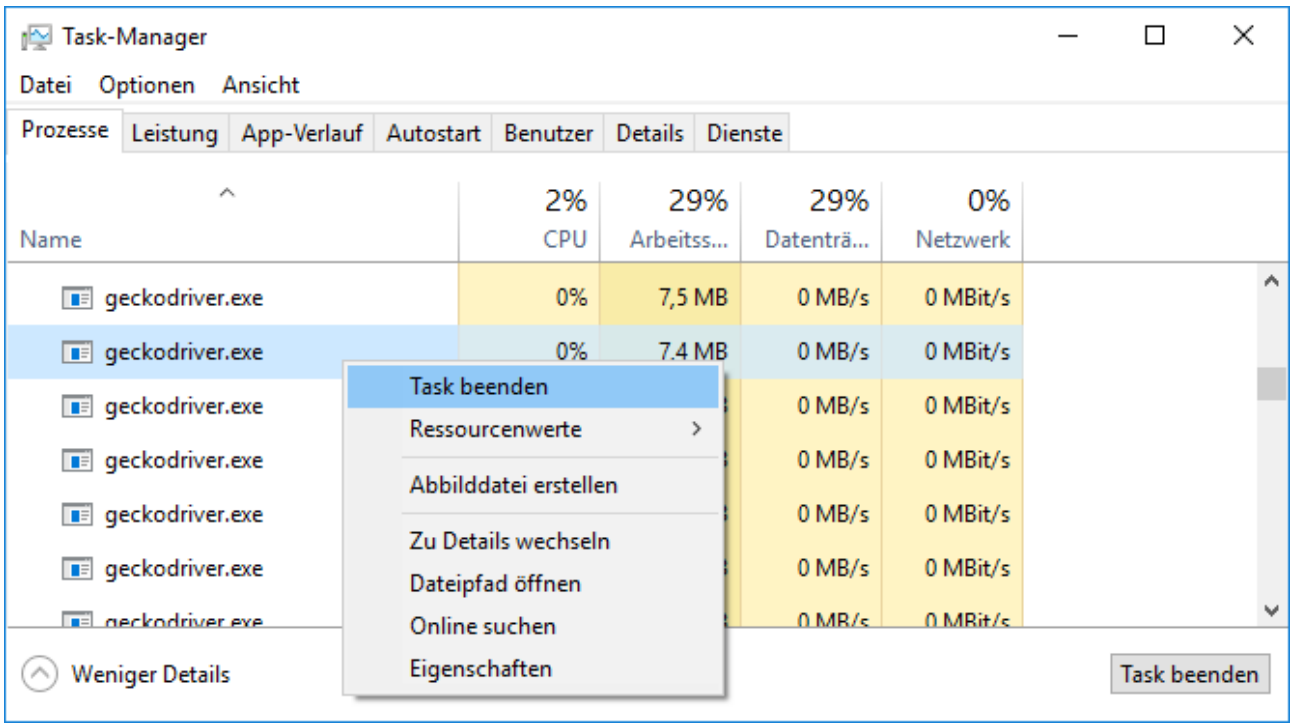
Sollte es Probleme mit der Zeile driver.quit(); geben, müssen die Fenster von Hand geschlossen werden. Weiterhin wird der verwendete Prozess geckodriver nicht geschlossen, was dann ebenfalls von Hand über den Task-Manager erfolgen muss. Dazu wird. Z. B. ein Rechtsklick an einer leeren Stelle in der Fußzeile gemacht und „Task-Manager“ ausgewählt.



Es wird unten auf „Mehr Details“ geklickt.



Es wird jeweils ein Rechtsklick auf „geckodriver.exe“ gemacht und „Task Beenden“ gewählt.



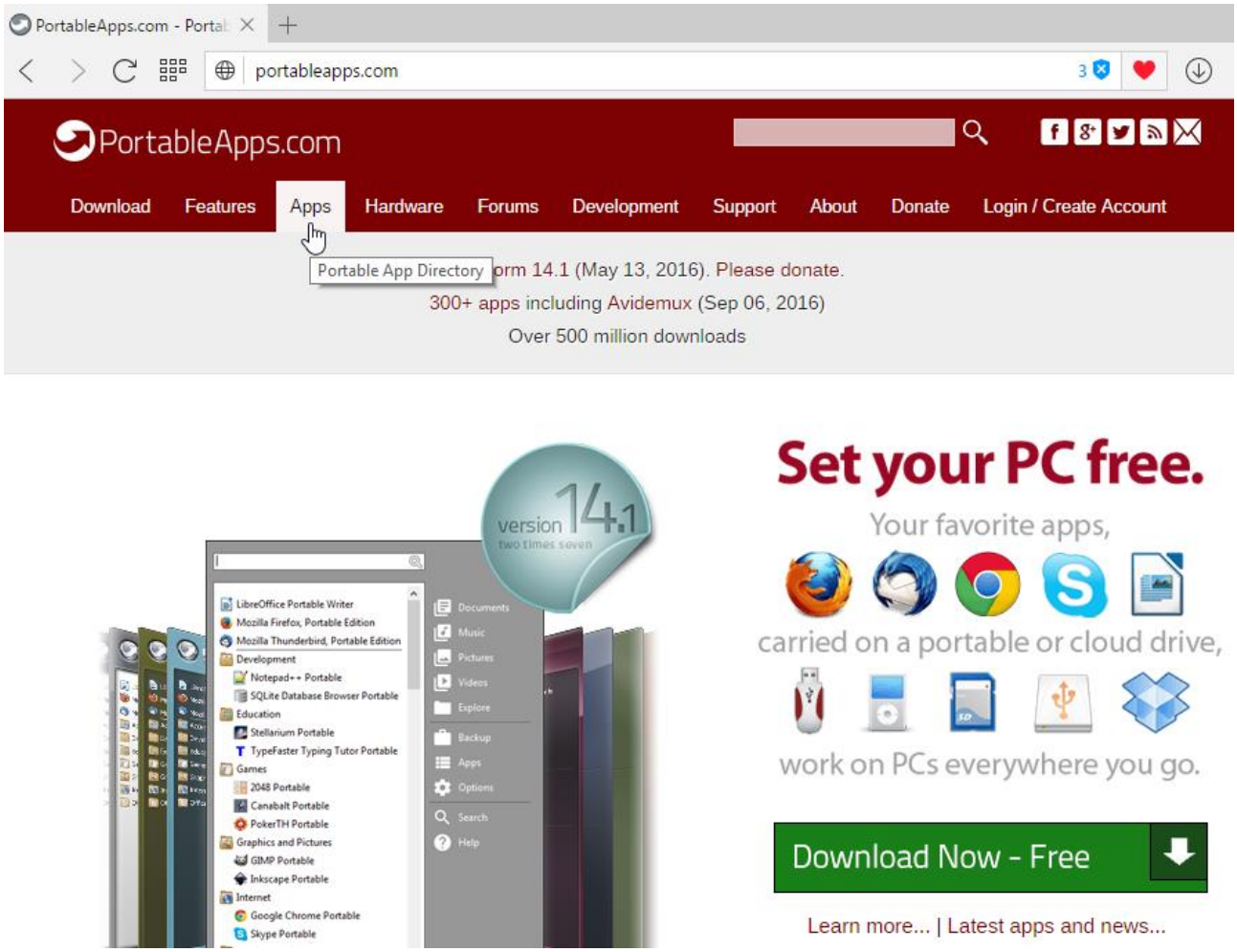
12.3 Selenium WebDriver mit portabler Browser-Installation

Soll flexibel eventuell sogar mit unterschiedlichen Firefox-Varianten experimentiert werden, bietet sich eine portable Version an, die dann lokal ihre Einstellungen enthält. Am Anfang der Selenium 3-Entwicklung lief dieser Ansatz zwischenzeitlich nicht auf allen Rechnertypen. In der vorgestellten Variante kann die setup-Methode, die den direkten Pfad zur ausführbaren Datei enthält, wie folgt aussehen.

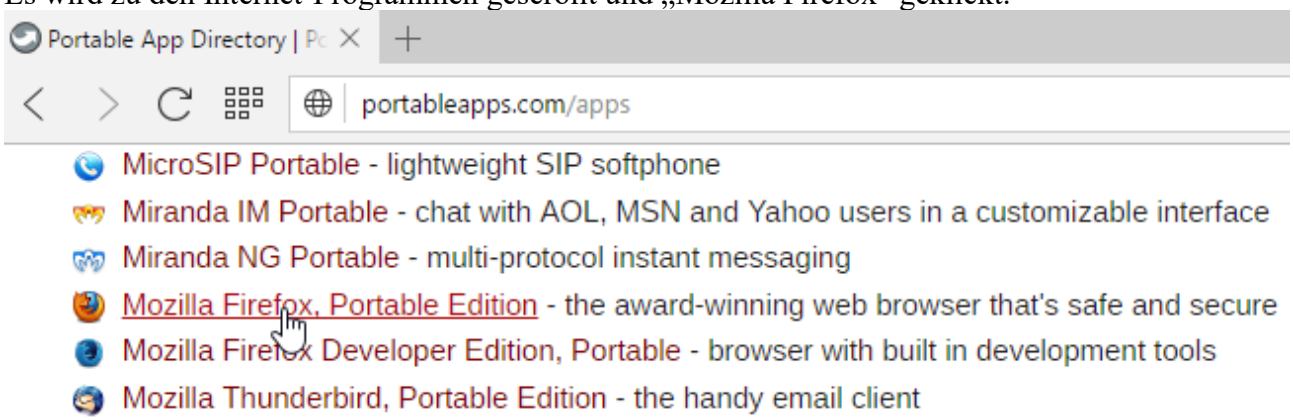
```
@Before
public void setUp() {
    DesiredCapabilities capabilities = DesiredCapabilities.firefox();
    capabilities.setCapability("marionette", true);
    FirefoxProfile profile = new FirefoxProfile();
    File exe = new File("F:\\Programme\\netbeans\\FirefoxPortable\\FirefoxPortable.exe");
    System.out.println(exe.exists() + " " + exe.canExecute());
    driver = new FirefoxDriver( new FirefoxBinary(exe) ,profile, capabilities);

    //driver = new FirefoxDriver(capabilities);
    //driver = new MarionetteDriver(); // veraltet
}
```

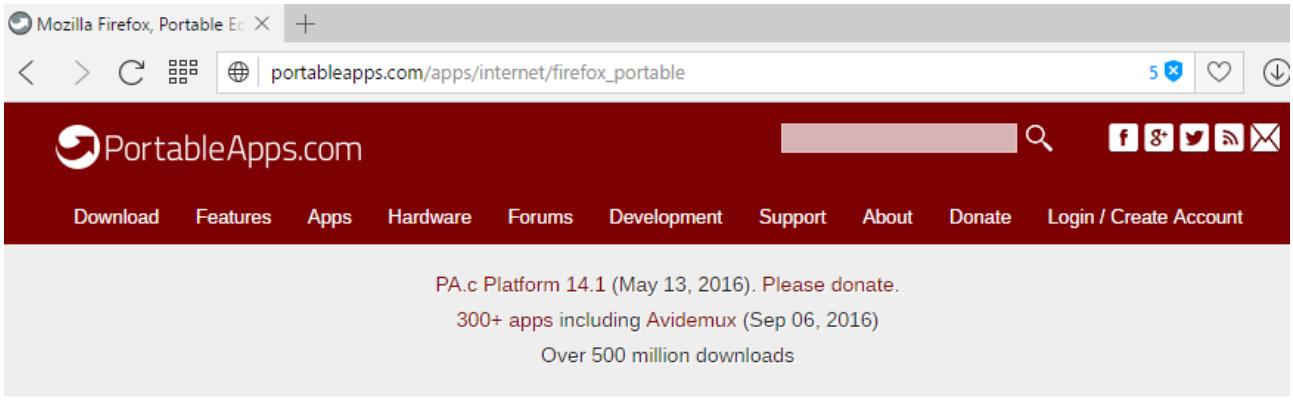
Die portable Version ist z. B. unter portableApps.com http://portableapps.com erhältlich, es wird auf „Apps“ geklickt.

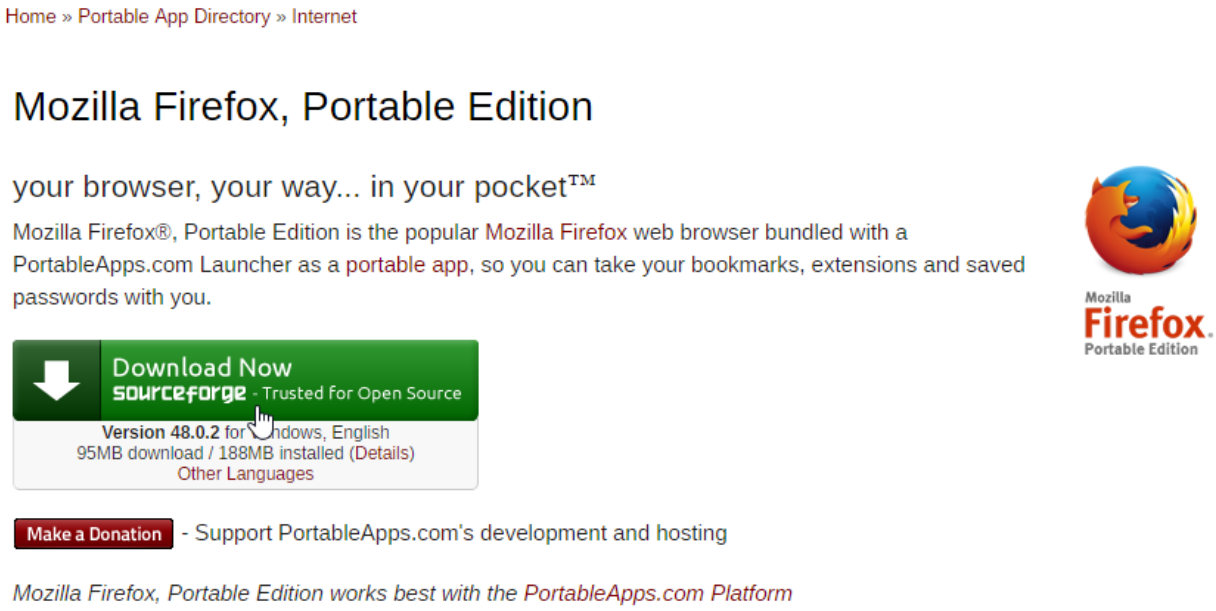


Es wird zu den Internet-Programmen gescrollt und „Mozilla Firefox“ geklickt.

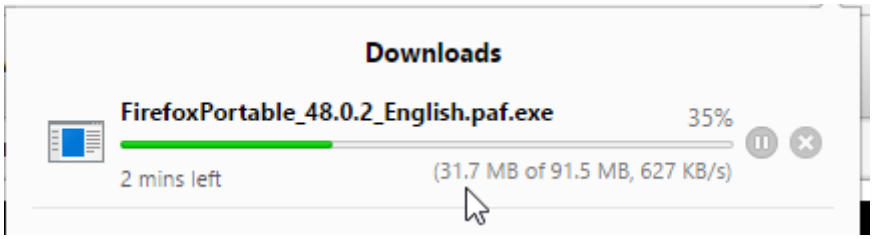


Es wird der Download-Button geklickt.

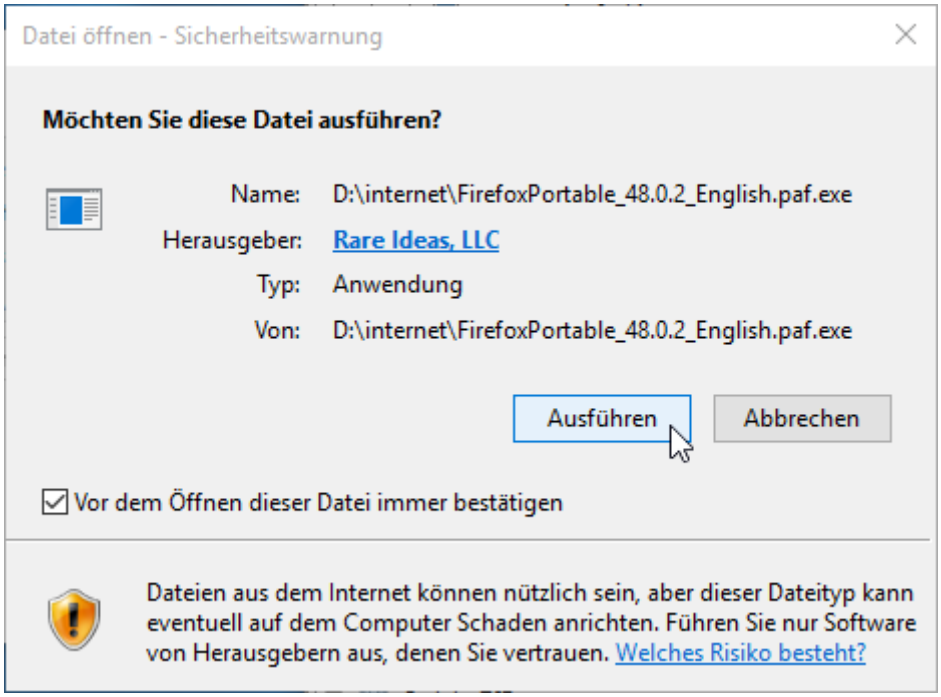




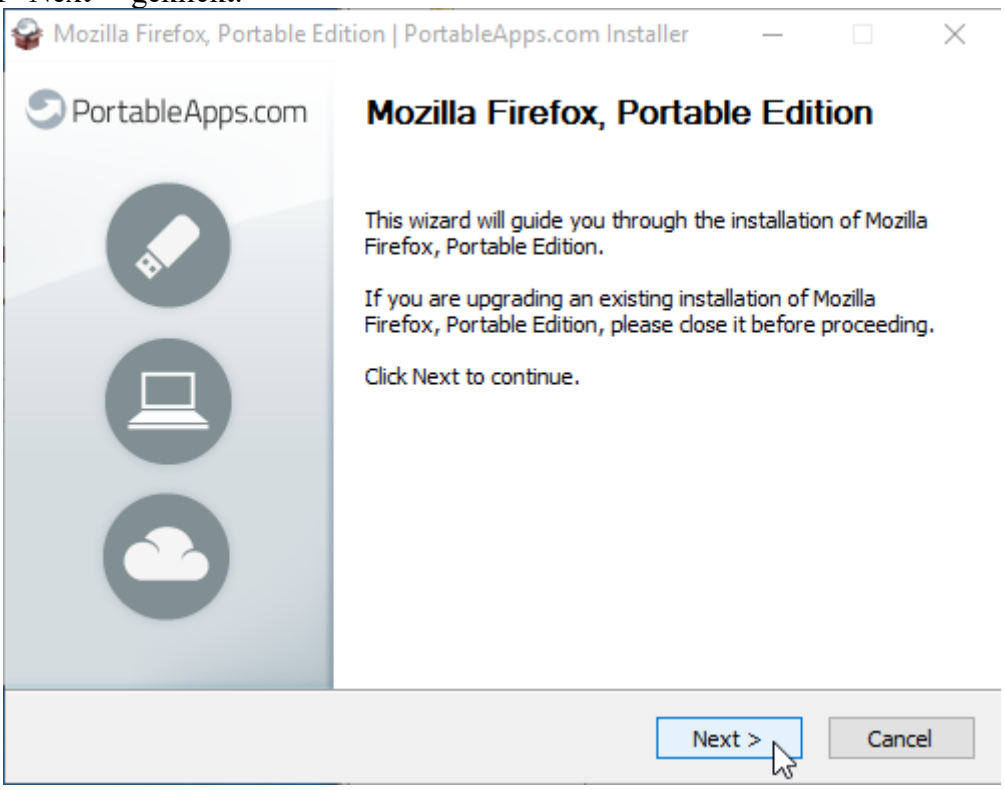
Nach kurzem Warten startet der Download.



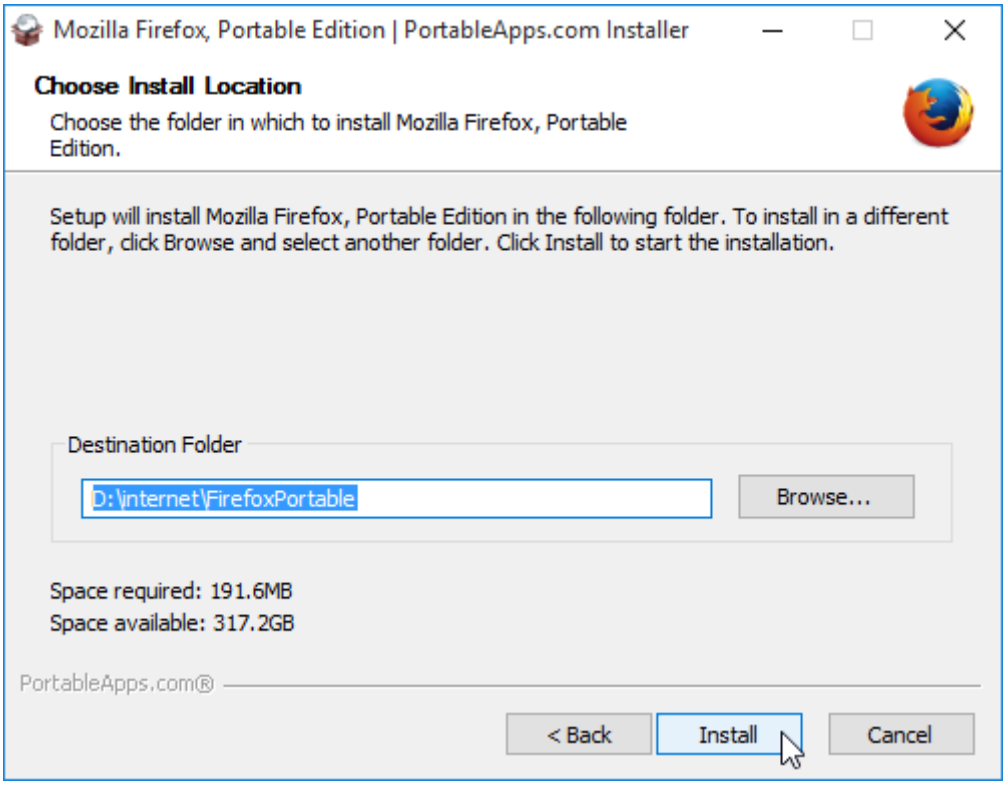
Die Datei wird ausgeführt.



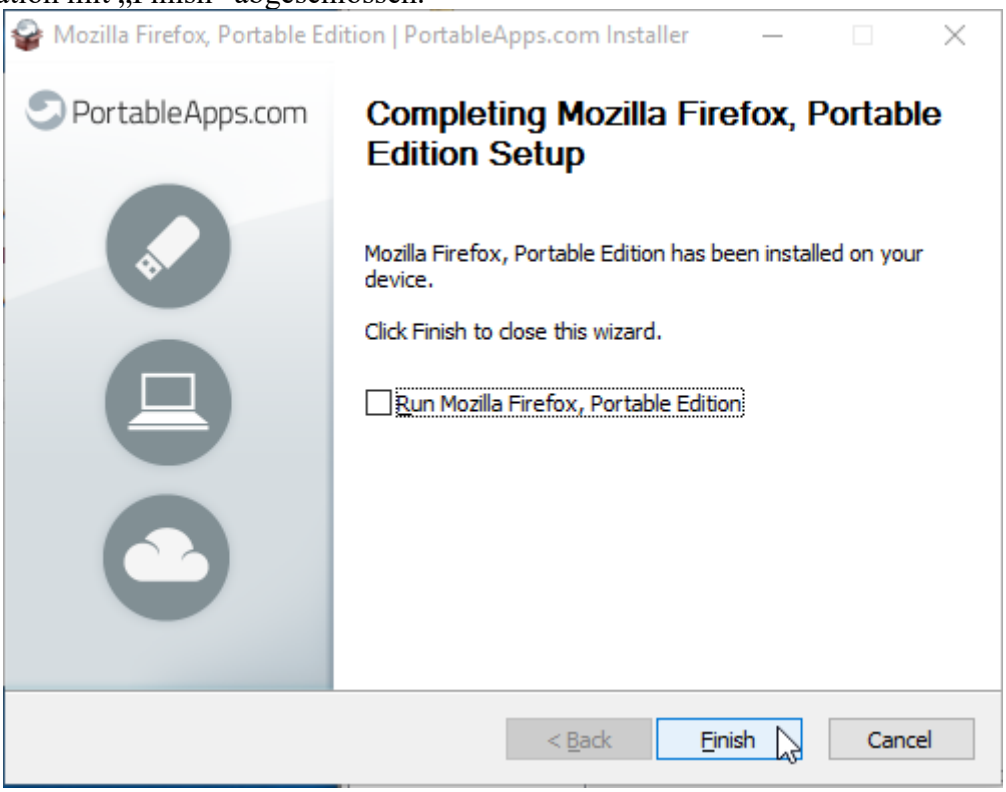
Es wird auf “Next>“ geklickt.



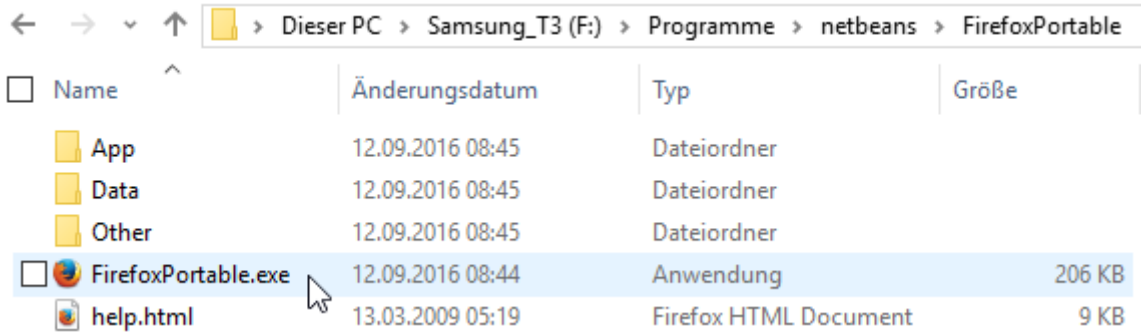
Der Zielordner mit „Install“ bestätigt, dieser wird nur temporär genutzt.



Die Installation mit „Finish“ abgeschlossen.

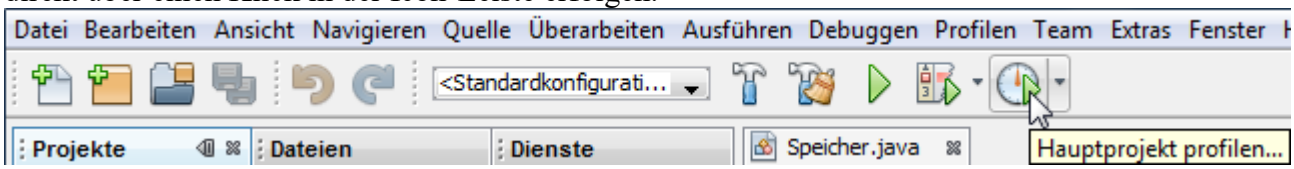


Der neue entstandene Ordner wird dann an eine beliebige Stelle geschoben, an der Programme ausführbar sind. Der Ordner enthält die zum Starten benötigte Datei.

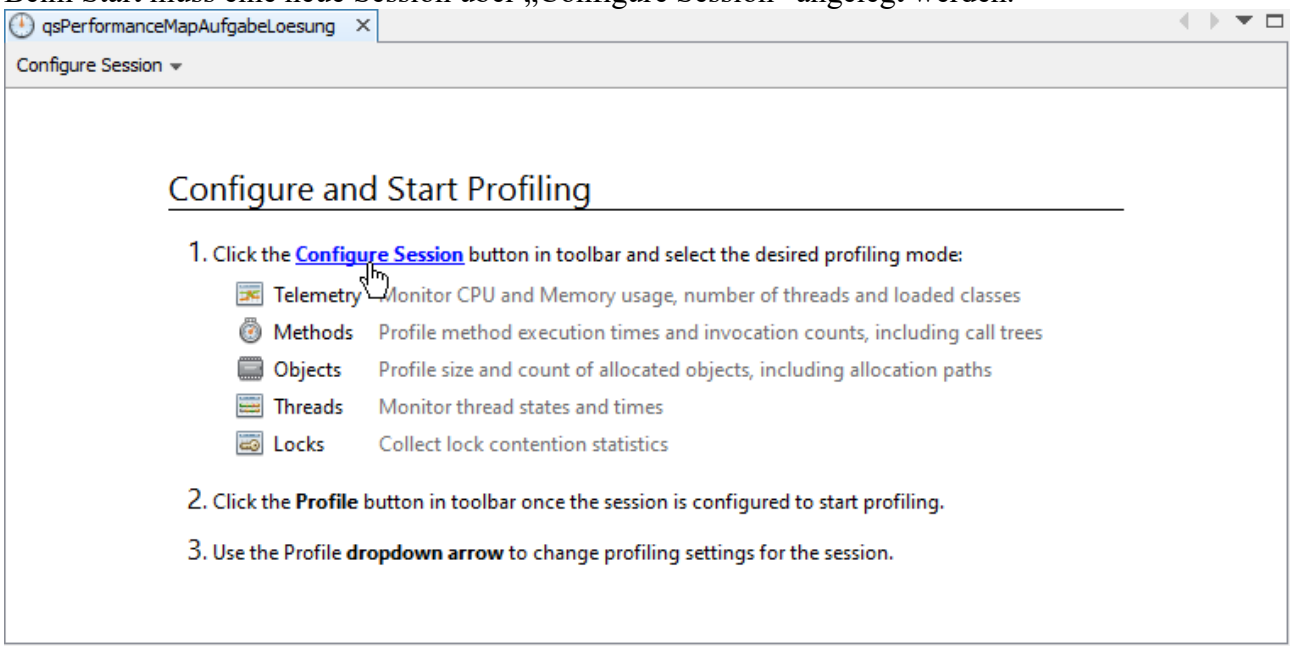


13 Einführung in den Profiler

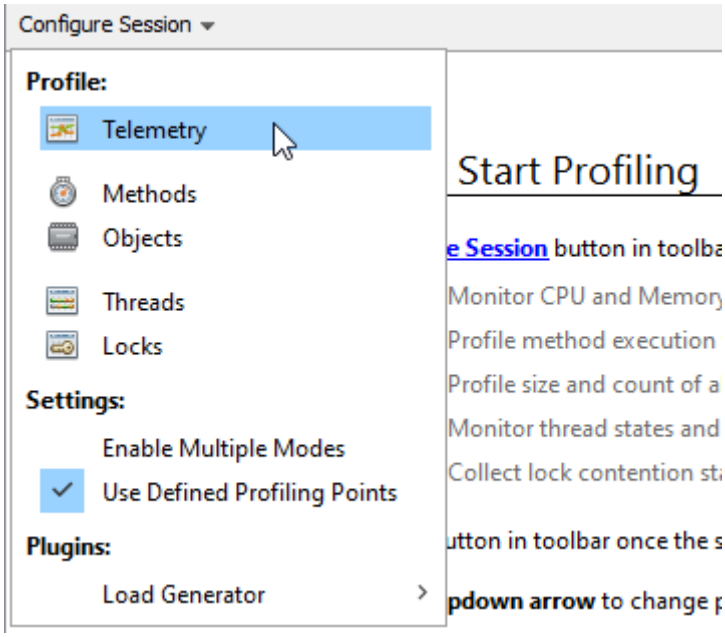
Java unterstützt von Haus aus einfache Performance-Messungen (Anzahl und Dauer der Methodenaufrufe), die durch den Profiler ausgewertet werden können. Der Start des Profilers kann direkt über einen Klick in der Icon-Leiste erfolgen.



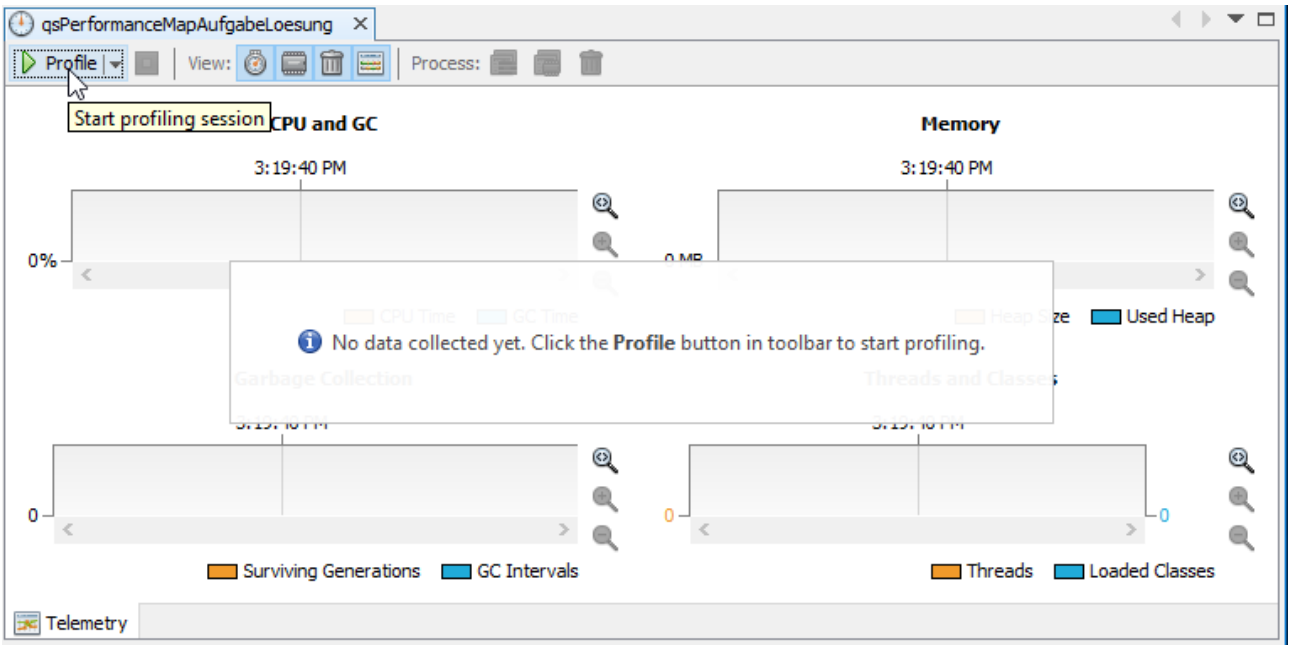
Beim Start muss eine neue Session über „Configure Session“ angelegt werden.



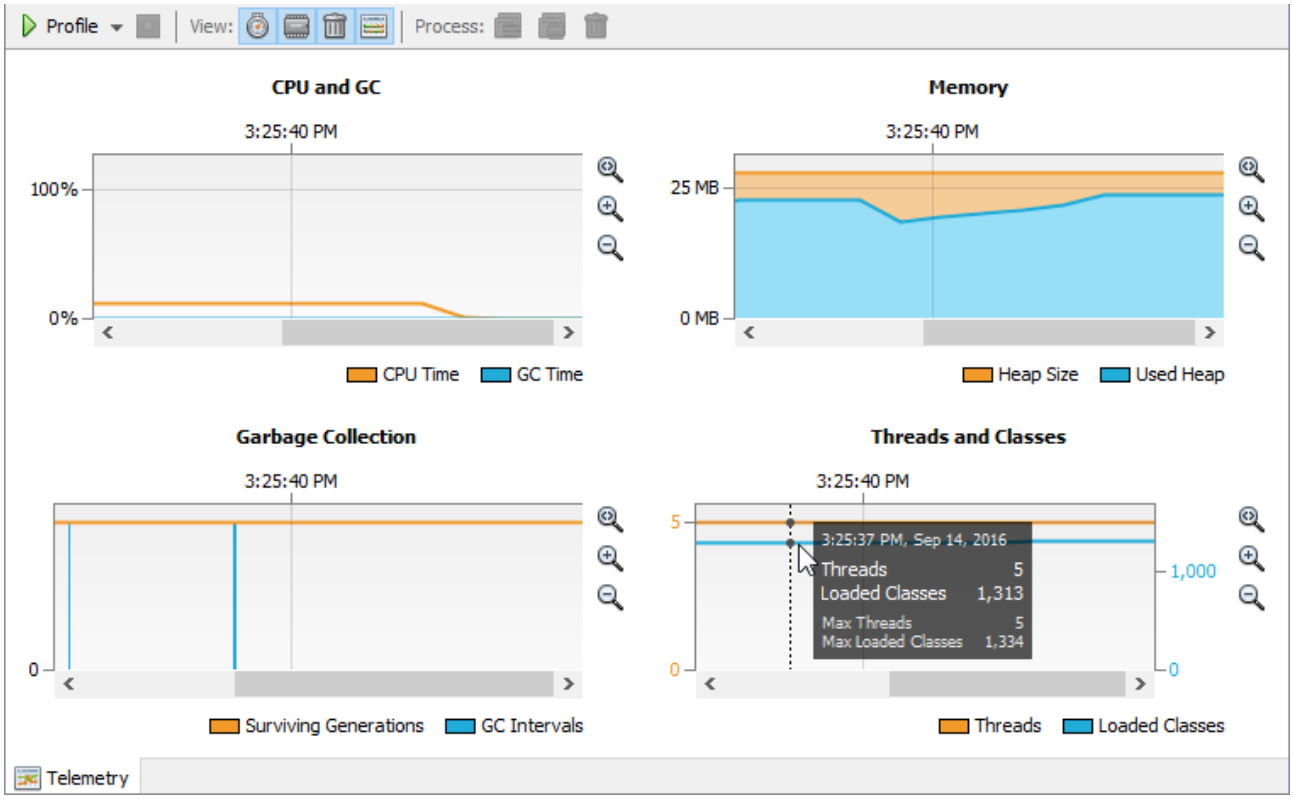
Es gibt mehrere Einstellungen die hier schrittweise vorgestellt werden. Die erste Auswahlmöglichkeit ist Telemetry.



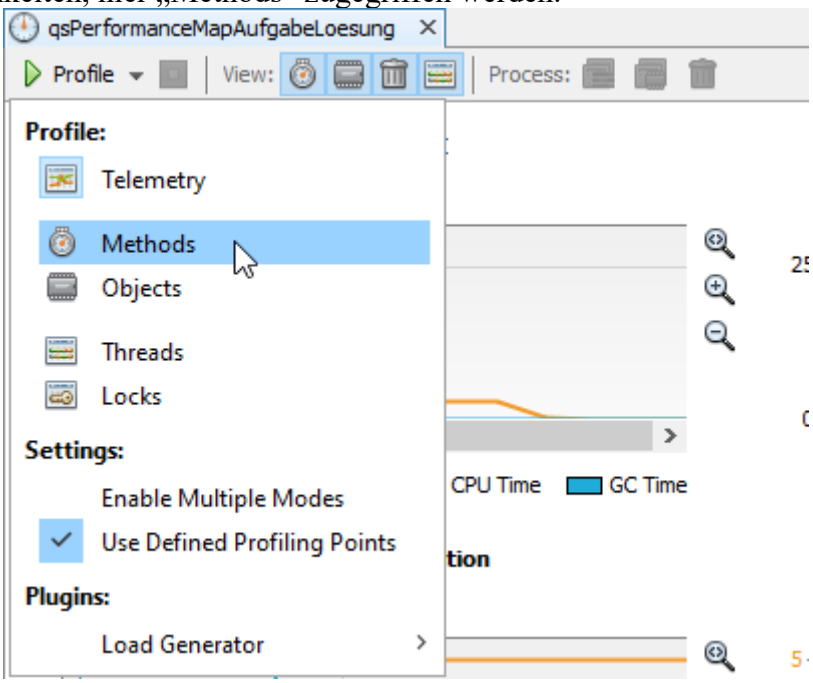
Es werden hier mehrere Kurven zur Nutzung des Speichers und der Anzahl existierender Objekte angegeben. Der Start erfolgt mit einem Klick auf den kleinen grünen Pfeil links oben. Etwas weiter rechts befinden sich Toggle-Button mit denen die einzelnen Anzeigen ein- und ausgeschaltet werden können.



Nach einem Programmdurchlauf kann die Maus in den einzelnen Diagrammen platziert werden, wodurch es zu Anzeige detaillierterer Informationen kommt.

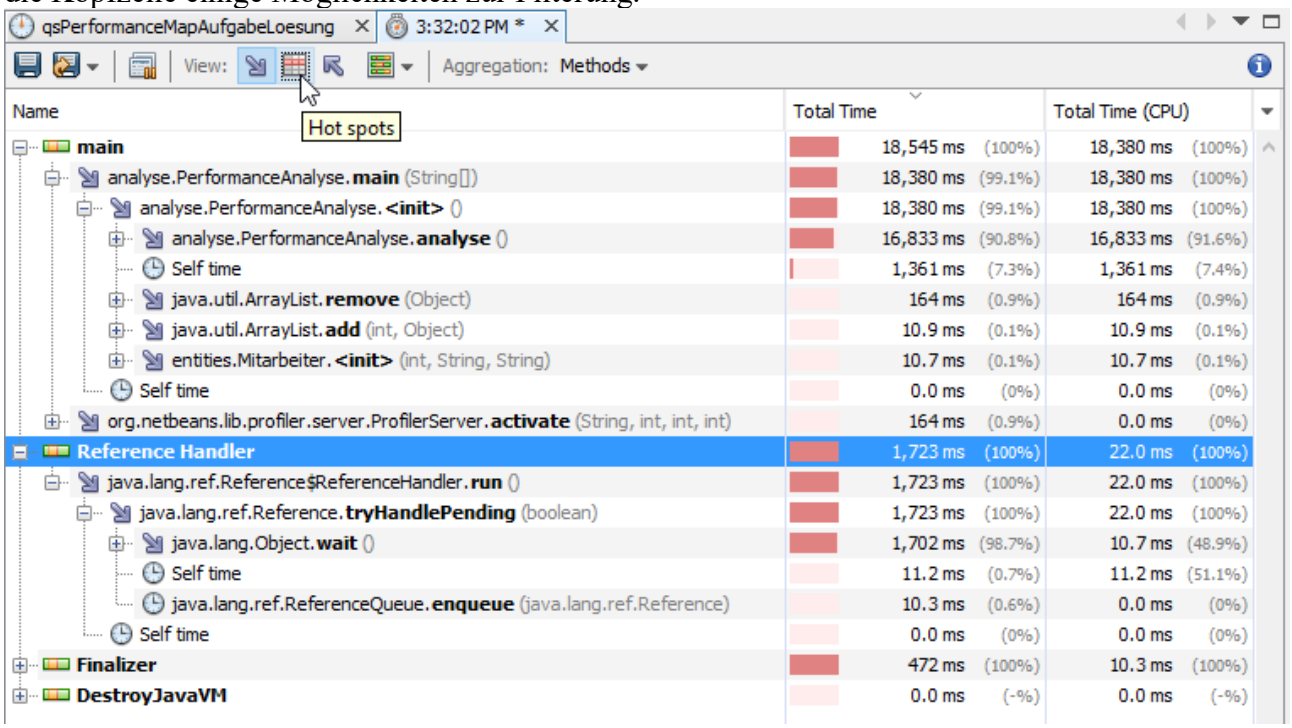


Wird das Menü zum kleinen Pfeil nach unten neben dem Wort „Profile“ aufgeklappt kann auf die weiteeren Möglichkeiten, hier „Methods“ zugegriffen werden.

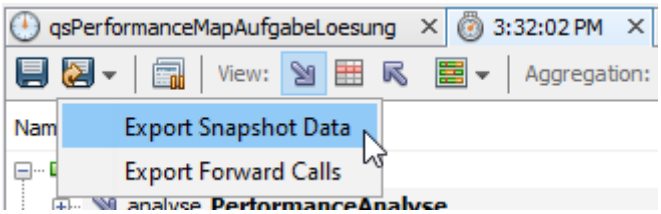


Die Ausführung wird wieder über einen Klick auf „Profile“ gestartet. Zur Ergebnisbearbeitung bietet

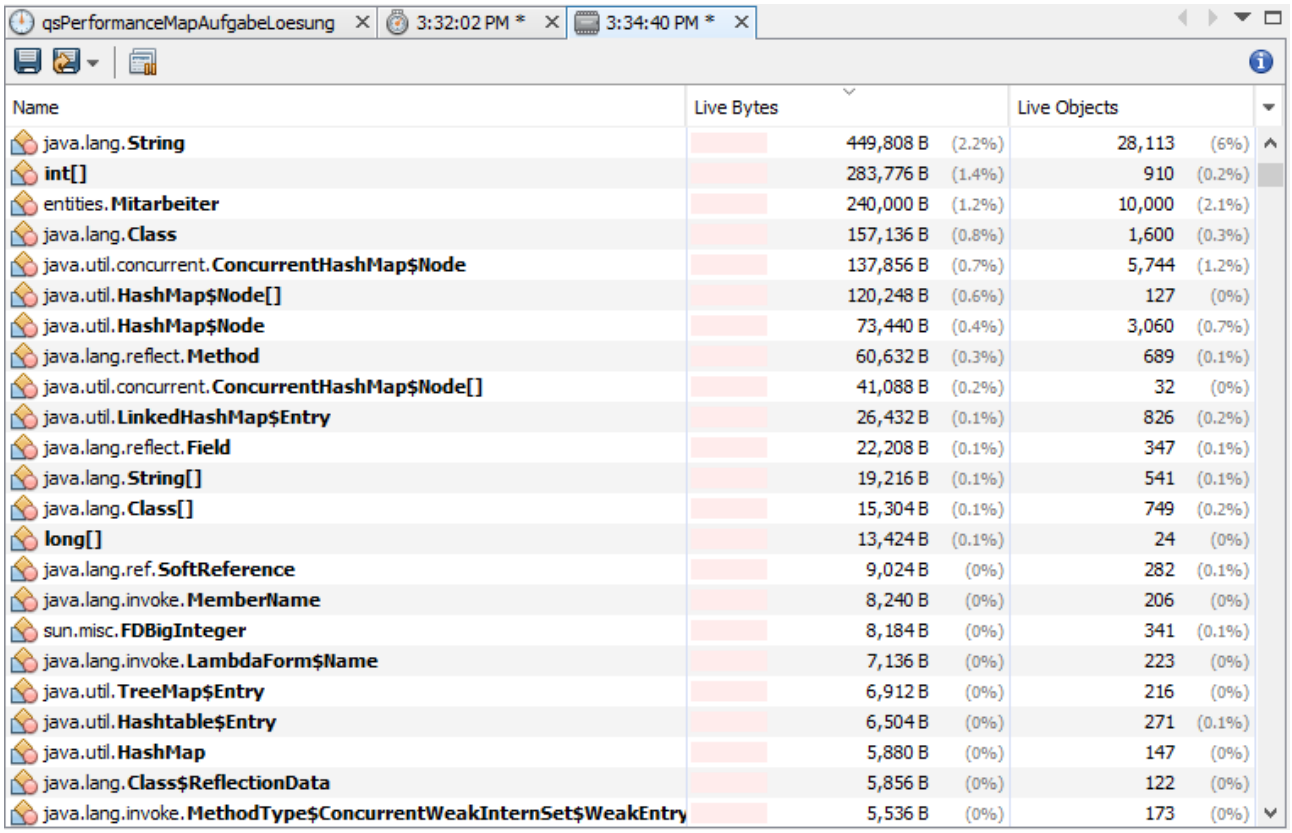
die Kopfzeile einige Möglichkeiten zur Filterung.



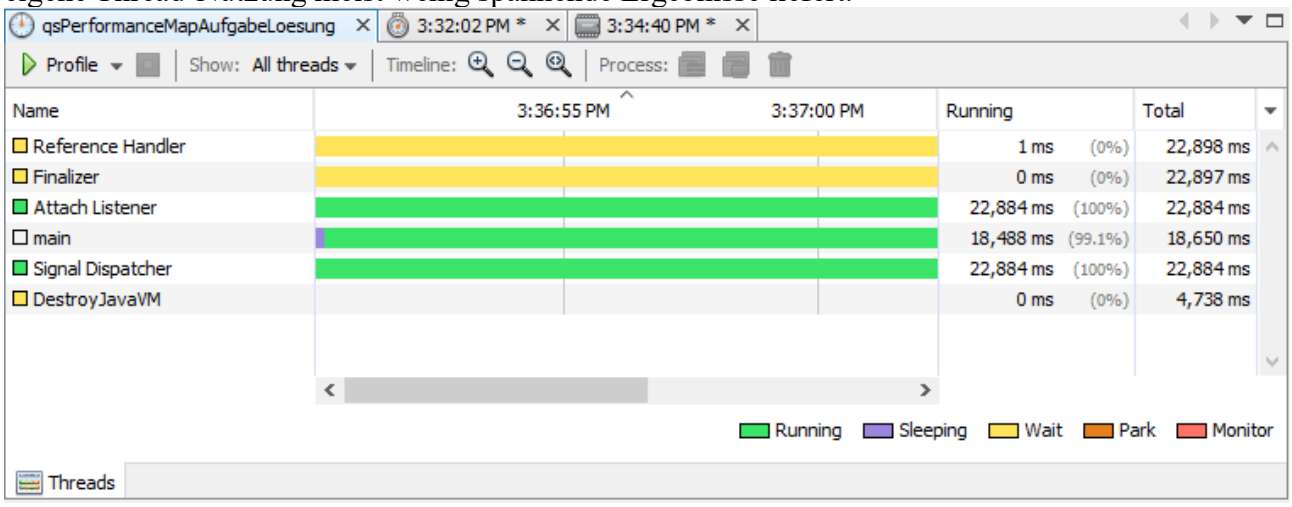
Die gemessenen Ergebnisse sind im Projekt speicherbar und über das zweite Diskettensymbol exportierbar.



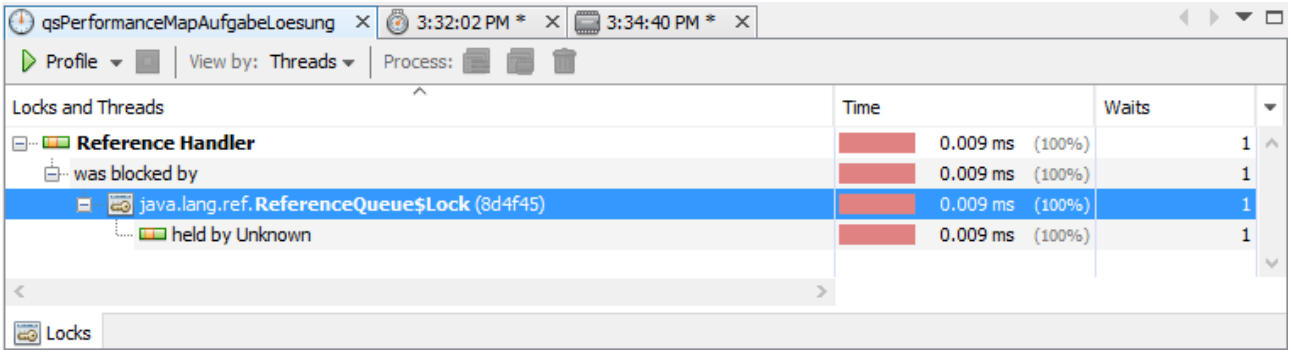
Mit der Auswahl „Objects“ ist die Anzahl der erstellten Objekte und deren Speicherverbrauch analysierbar.



Mit „Threads“ werden die genutzten Threads aufgezeigt, was für Programme ohne Oberfläche und eigene Thread-Nutzung meist wenig spannende Ergebnisse liefert.



Mit „Locks“ wird erkannt, ob und wann gewisse Objekte als Sperren genutzt werden, um einen gleichzeitigen Zugriff zu verhindern.

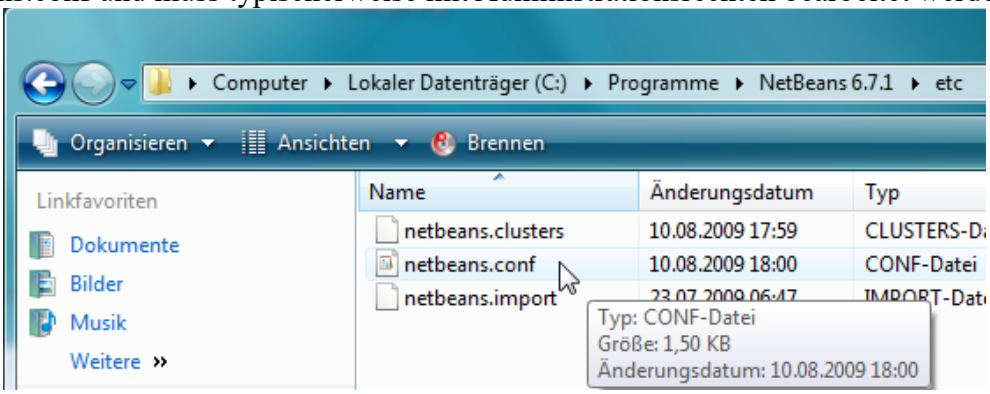

qsPerformanceMapAufgabeLoesung
3:32:02 PM *
3:34:40 PM *
Profile
View by: Threads
Process:
Locks and Threads
Time
Waits
Reference Handler
0.009 ms (100%)
1
was blocked by
0.009 ms (100%)
1
java.lang.ref.ReferenceQueue$Lock (8d4f45)
0.009 ms (100%)
1
held by Unknown
0.009 ms (100%)
1
Locks

14 Tipps und Fragen

14.1 Sehr langsame Übersetzung von Web-Projekten

Bei Web-Projekten mit Server-Einsatz und Datenbanknutzung kann ein älterer Rechner sehr schnell sehr langsam werden. Eine generelle Abhilfe gibt es außer einem Rechnerkauf leider nicht, trotzdem können folgende Schritte die Arbeit etwas erleichtern.

1. Man kann in der Startdatei von NetBeans einige Dinge konfigurieren. Die Datei findet man im Installationsordner von NetBeans im Unterverzeichnis etc. Sie hat den Namen NetBeans.conf und muss typischerweise mit Administrationsrechten bearbeitet werden.



   Auf den Web-Seiten zu NetBeans findet man einige Anregungen. Plump beschrieben, geben Zahlen mit kleinem „m“ dahinter Speichergrößen an, die man z. B. verdoppeln (ver n-fachen) kann. Die wichtigsten Werte stehen unter „# command line switches“. Das folgende Bild zeigt die Ausgangseinstellungen.

```
# ${HOME} will be replaced by JVM user.home system property
netbeans_default_userdir="${HOME}/.netbeans/6.7"

# Options used by NetBeans launcher by default, can be overridden by explicit
# command line switches:
netbeans_default_options="-J-Dcom.sun.aas.installRoot=\"C:\Sun\AppServer\" -J-client -J-Xss2m
-J-Xms32m -J-XX:PermSize=32m -J-XX:MaxPermSize=200m -J-Xverify:none -J-
Dapple.laf.useScreenMenuBar=true -J-Dsun.java2d.noddraw=true"
# Note that a default -Xmx is selected for you automatically.
# You can find this value in var/log/messages.log file in your userdir.
# The automatically selected value can be overridden by specifying -J-Xmx here
# or on the command line.

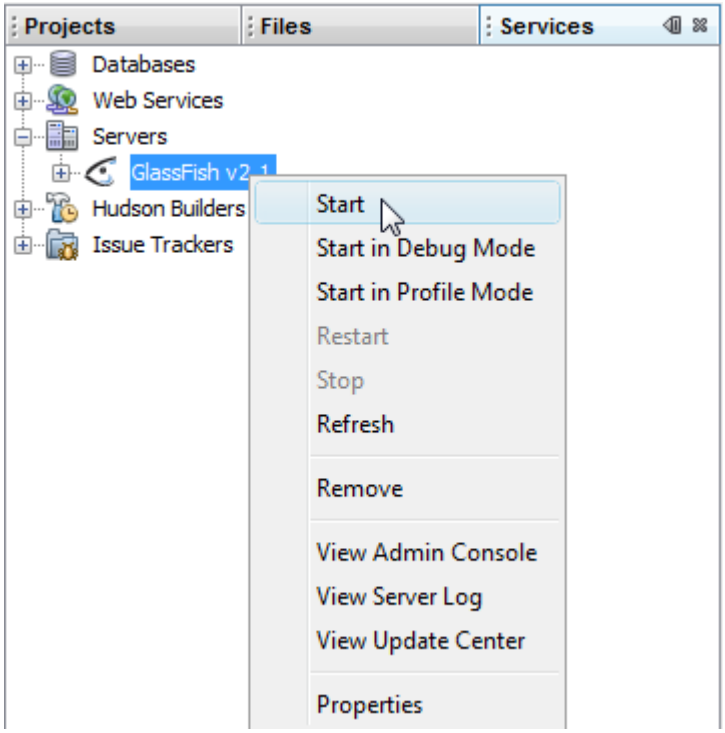
# If you specify the heap size (-Xmx) explicitely, you may also want to enable
# Concurrent Mark & Sweep garbage collector. In such case add the following
# options to the netbeans_default_options:
# -J-XX:+UseConcMarkSweepGC -J-XX:+CMSClassUnloadingEnabled -J-XX:+CMSPermGenSweepingEnabled
# (see http://wiki.netbeans.org/wiki/view/FaqGCPauses)

# Default location of JDK, can be overridden by using --jdkhome <dir>:
netbeans_jdkhome="C:\Program Files\Java\jdk1.6.0_12"

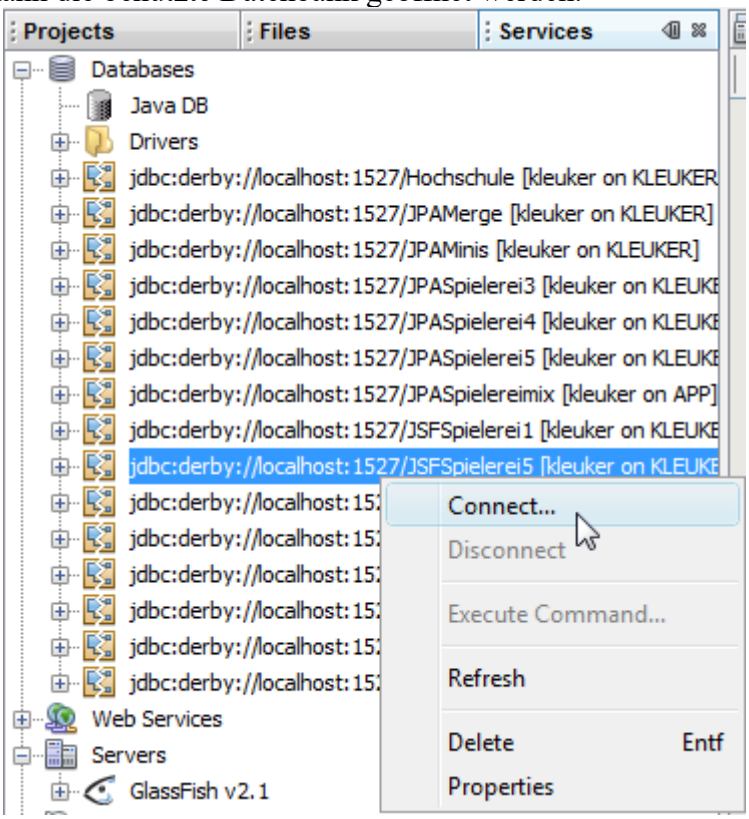
# Additional module clusters, using ${path.separator} (';' on Windows or ':' on Unix):
#netbeans_extraclusters="/absolute/path/to/cluster1:/absolute/path/to/cluster2"

# If you have some problems with detect of proxy settings, you may want to enable
# detect the proxy settings provided by JDK5 or higher.
# In such case add -J-Djava.net.useSystemProxies=true to the netbeans_default_options.
```

2. Schließen Sie alle anderen Programme, die nicht unmittelbar für die Entwicklung benötigt werden, da diese hier sehr wichtigen Hauptspeicher belegen. Skype und Messenger lenken eh nur ab.
3. Ein Neustart von NetBeans führt dazu, dass nicht frei gegebene Ressourcen, unter anderem auch durch eigene unsaubere Programmierung, wieder freigegeben werden. Ein neues Projekt sollte bei einem langsamen System mit einem Neustart von NetBeans verknüpft werden.
4. Generell kann man den Server und die benutzten Datenbanken starten und öffnen, bevor das eigentliche Projekt übersetzt wird. Der Serverstart ist unter dem „GlassFish-Eintrag“ unter dem Reiter „Services“ möglich.

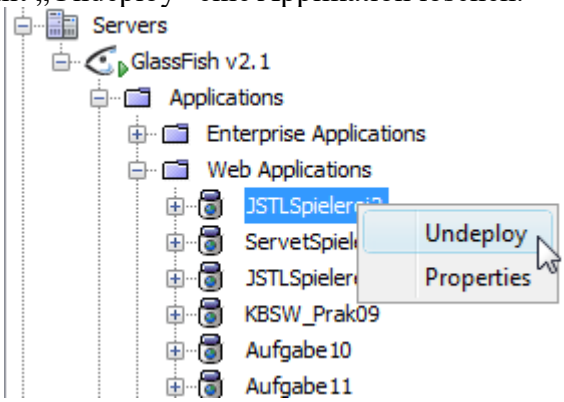


Unter Database kann die benutzte Datenbank geöffnet werden.

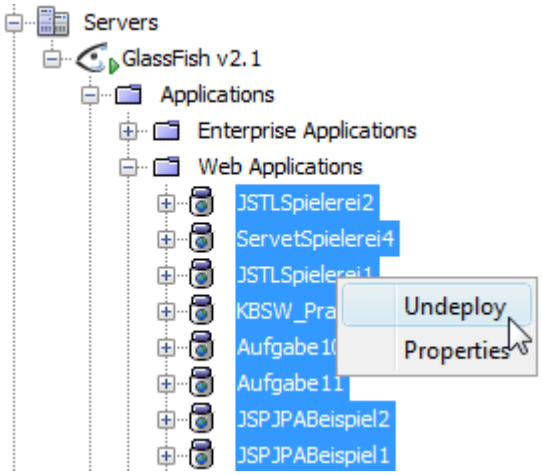


5. Der Server kann ebenfalls aufgeräumt werden, dies kann direkt im Administrationsbereich des Servers passieren oder ist auch unter NetBeans möglich. Wenn GlassFish läuft, kann man

sich unter „Web Applications“ die installierten Applikationen ansehen. Durch einen Rechtsklick kann man mit „Undeploy“ eine Applikation löschen.

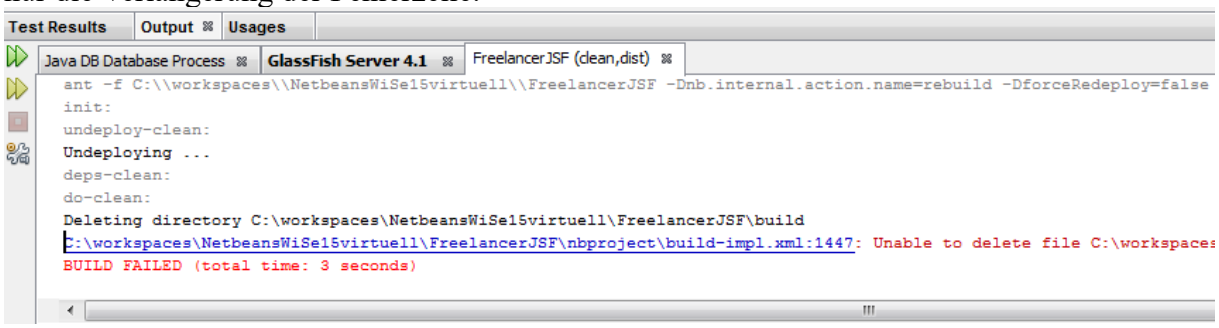


Mit gedrückter Strg-Taste können auch gleichzeitig mehrere Applikationen selektiert und dann entfernt werden.



14.2 Meldung „BUILD FAILED“

Wir ein Projekt mit „Clean and Build Project“ neu übersetzt , so kann es passieren, dass das Löschen einer Datei scheitert, wie in der folgenden Abbildung gezeigt. Die Abbildung danach zeigt nur die Verlängerung der Fehlerzeile.



Der Grund ist meist, dass eine der übersetzten Dateien (vermeintlich) noch von einem Prozess genutzt

wird. Hier kann es zumindest bei langsamen Rechnern sinnvoll sein, NetBeans neu zu starten. Alternativ kann man die Datei von Hand löschen. Hier werden drei Werkzeuge vorgeschlagen, die behilflich sein können. Wie immer gilt, dass man für sein Handeln selbst verantwortlich ist.

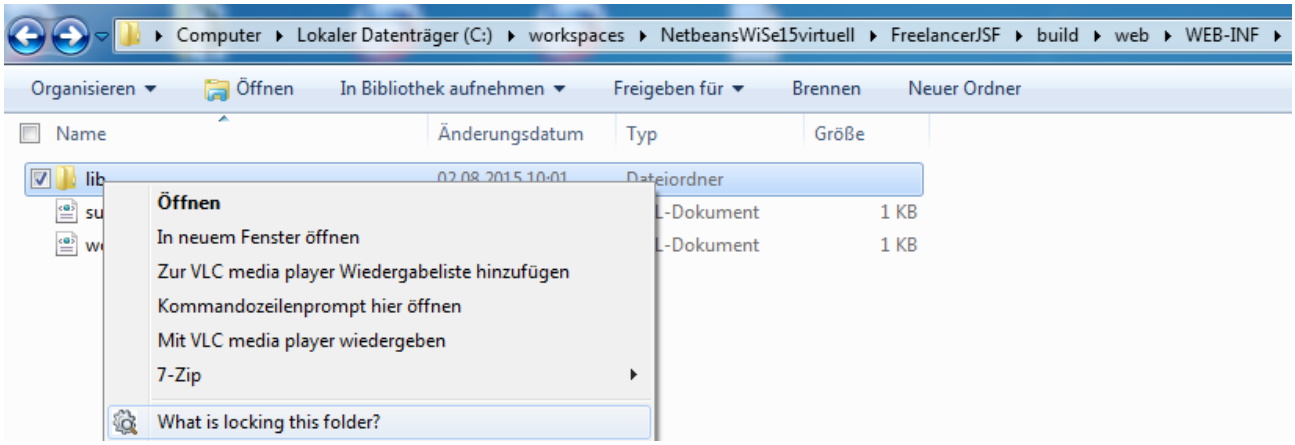
LockHunter
Falls Dateien oder Verzeichnisse nicht gelöscht werden können, da angeblich noch ein Prozess darauf zugreift, kann man diese Verbindung z. B. mit dem Programm Lockhunter lösen. Das Programm ist z. B. unter http://lockhunter.com/ oder http://www.chip.de/downloads/LockHunter_40730974.html erhältlich. Die Installation läuft so durch, es besteht die Möglichkeit den Installationspfad anzupassen.



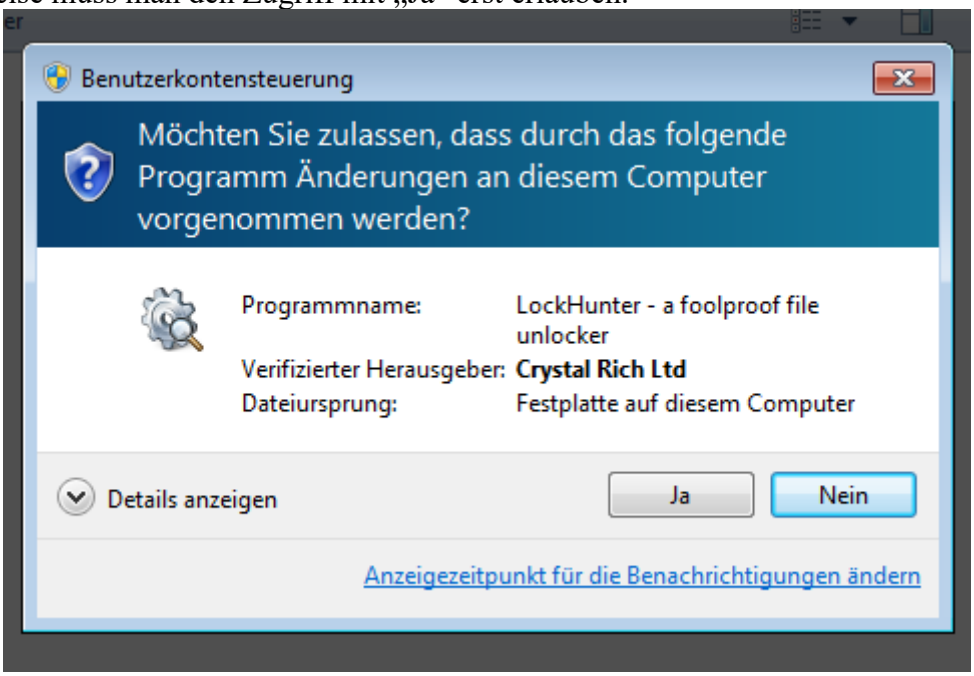
Zur Nutzung von LockHunter und den anderen Programmen wird zunächst in der Fehlermeldung der genaue Pfad ausgelesen.

```
Unable to delete file C:\workspaces\NetbeansWiSe15virtuell\FreelancerJSF\build\web\WEB-INF\lib\bson-3.0.2.jar
```

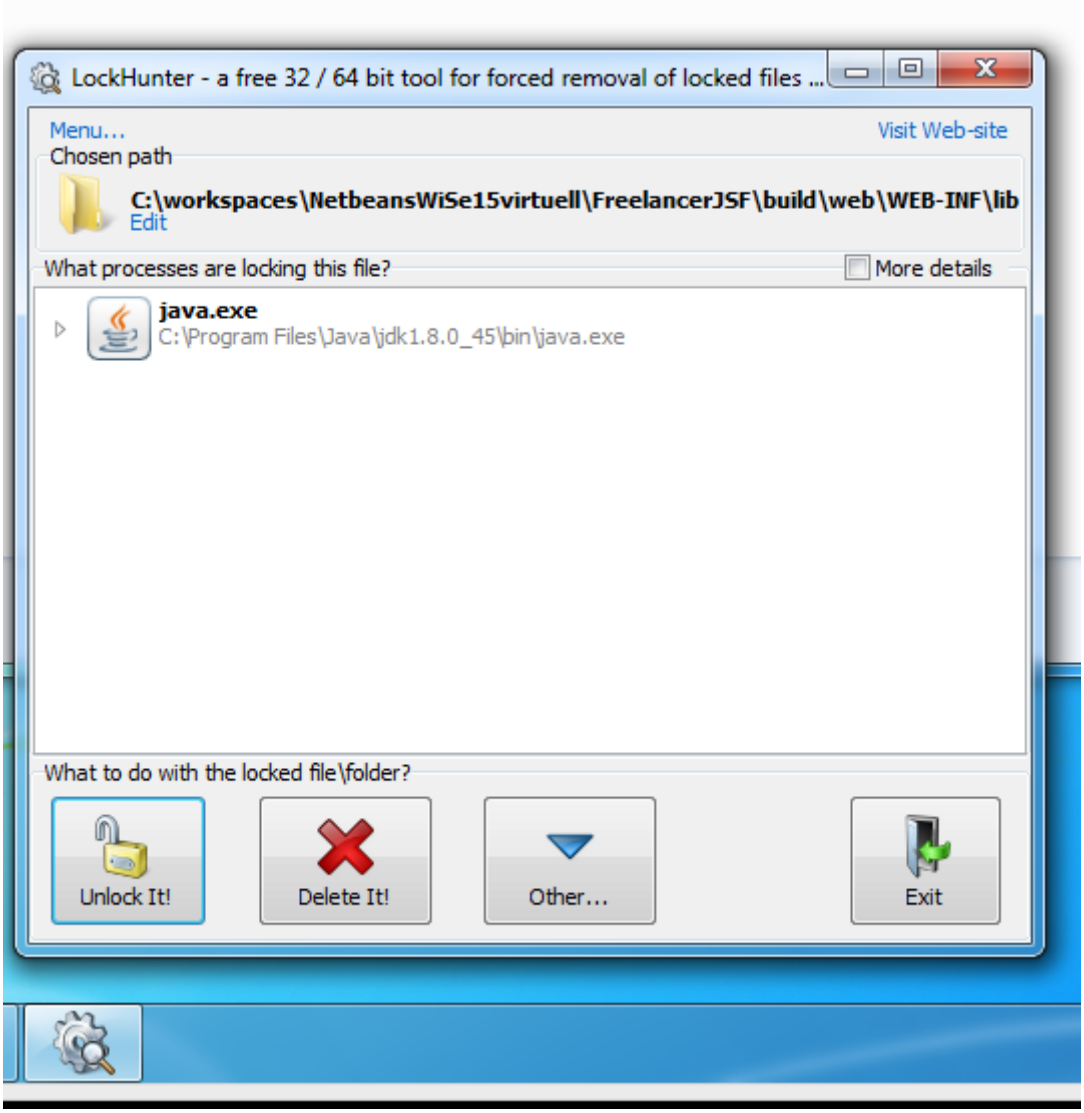
Dann wird mit dem Windows-Explorer zu dieser Datei oder dem darüber liegenden Verzeichnis gesteuert, ein Rechtsklick gemacht und „What is locking this folder?“ ausgewählt.



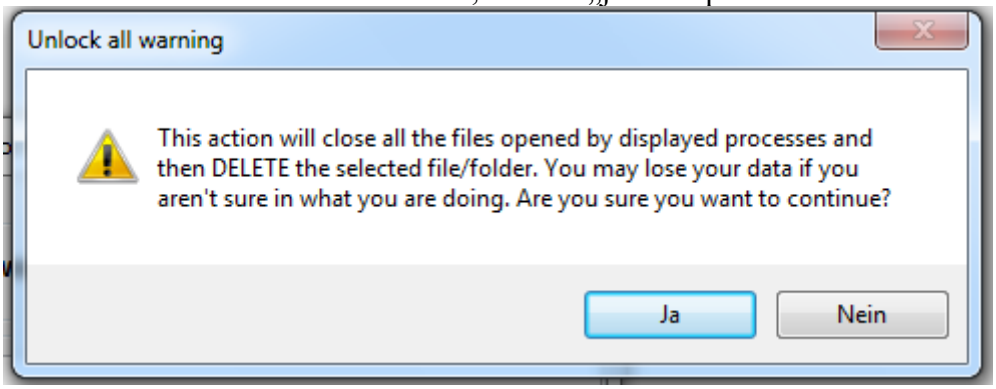
Typischerweise muss man den Zugriff mit „Ja“ erst erlauben.



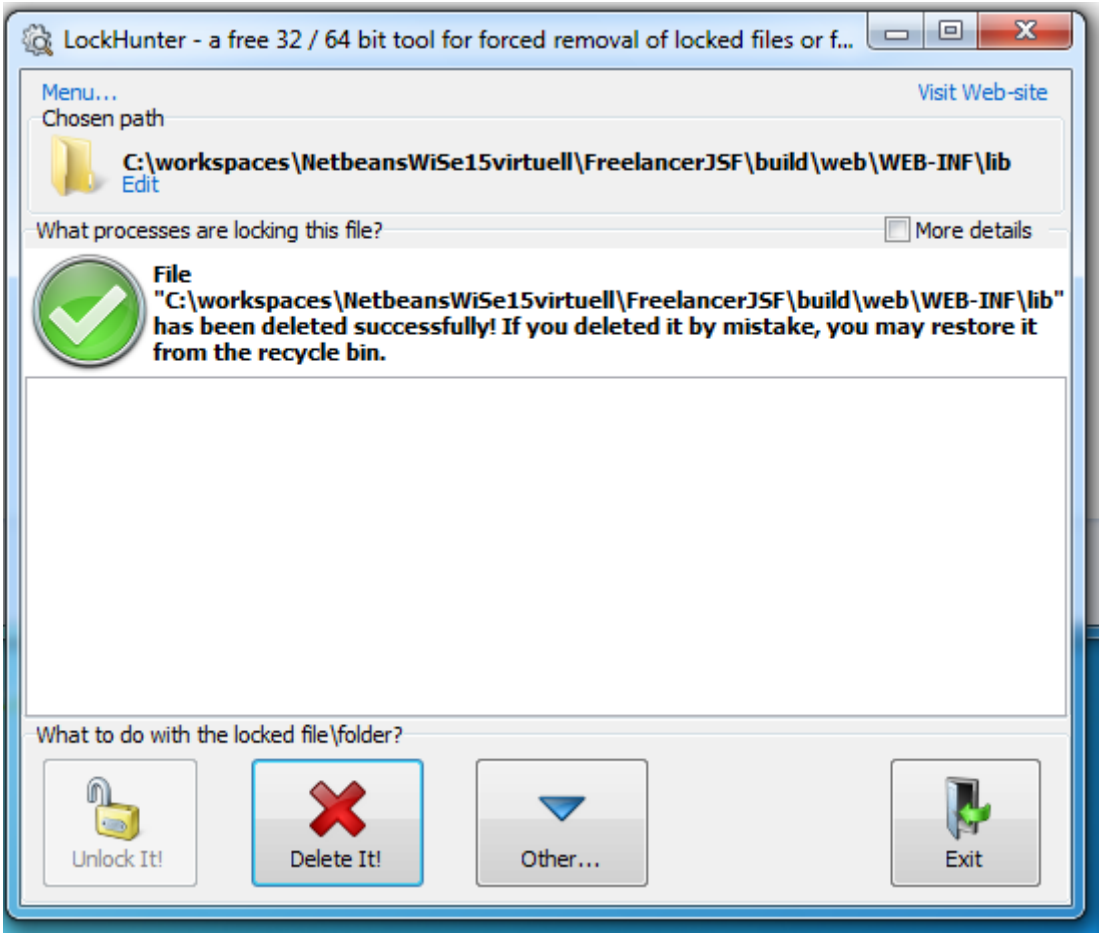
Danach kann man die Sperre entfernen oder einfach auf „Delete It!“ klicken. Es kann passieren, dass das zugehörige Fenster hinter anderen Fenstern liegt und erst in der Task-Leiste gesucht werden muss.



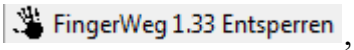
Es gibt einen Hinweis auf den Effekt der Aktion, der mit „ja“ akzeptiert wird.



Abschließend wird eine Information ausgegeben, dass der Schritt erfolgreich war. Das Programm wird mit „Exit“ verlassen. Im nächsten Schritt wird wieder „Clean and Build“ angestoßen.

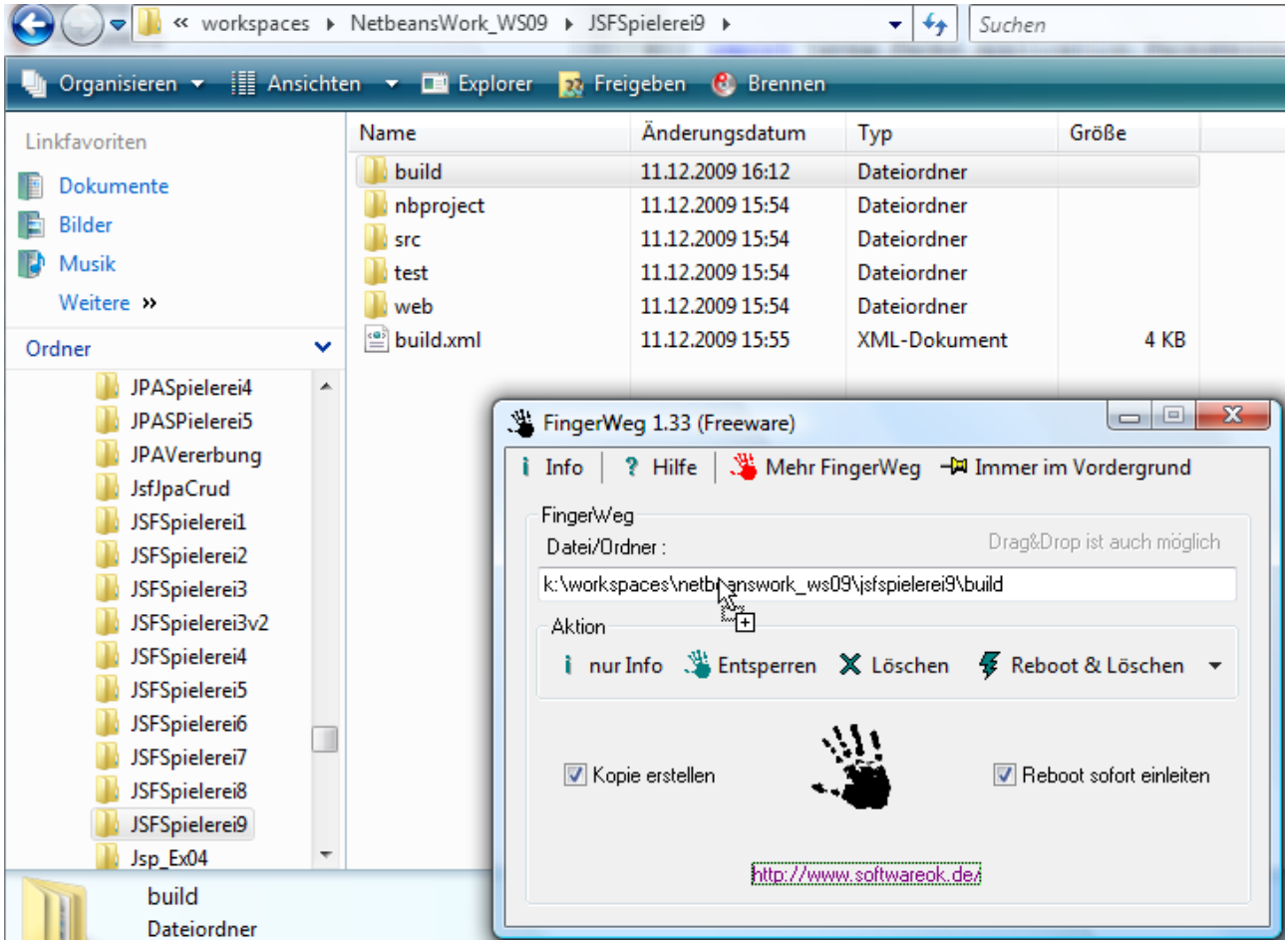


FingerWeg

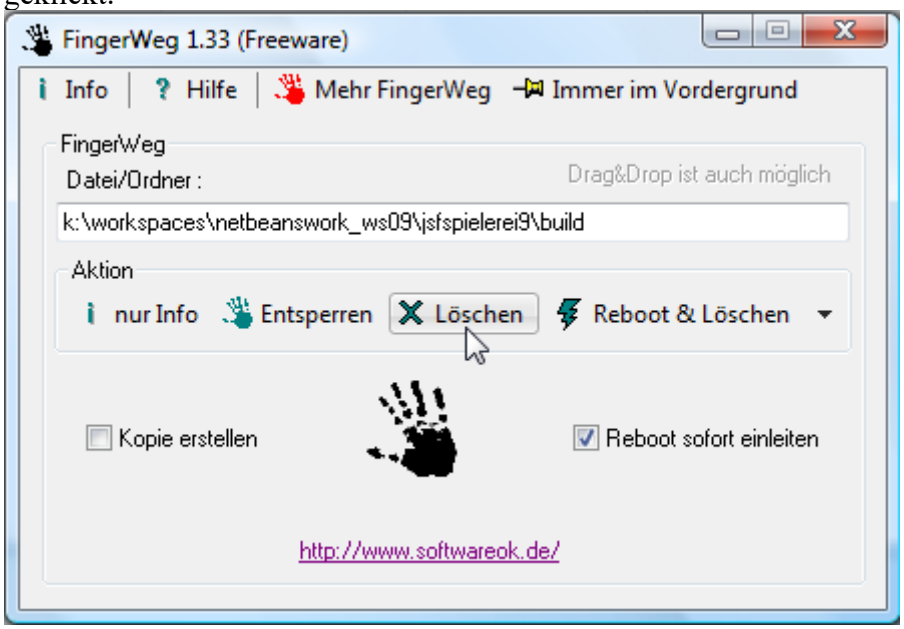
Eine Alternative ist z. B. das Werkzeug „FingerWeg“ 

, was z. B. unter http://www.softwareok.de/?seite=Freeware/FingerWeg erhältlich ist. Bei der Installation ist zu prüfen, ob nicht irgendwelche unerwünschte Zusatzsoftware installiert wird (änderte sich leider zwischen den Versionen).

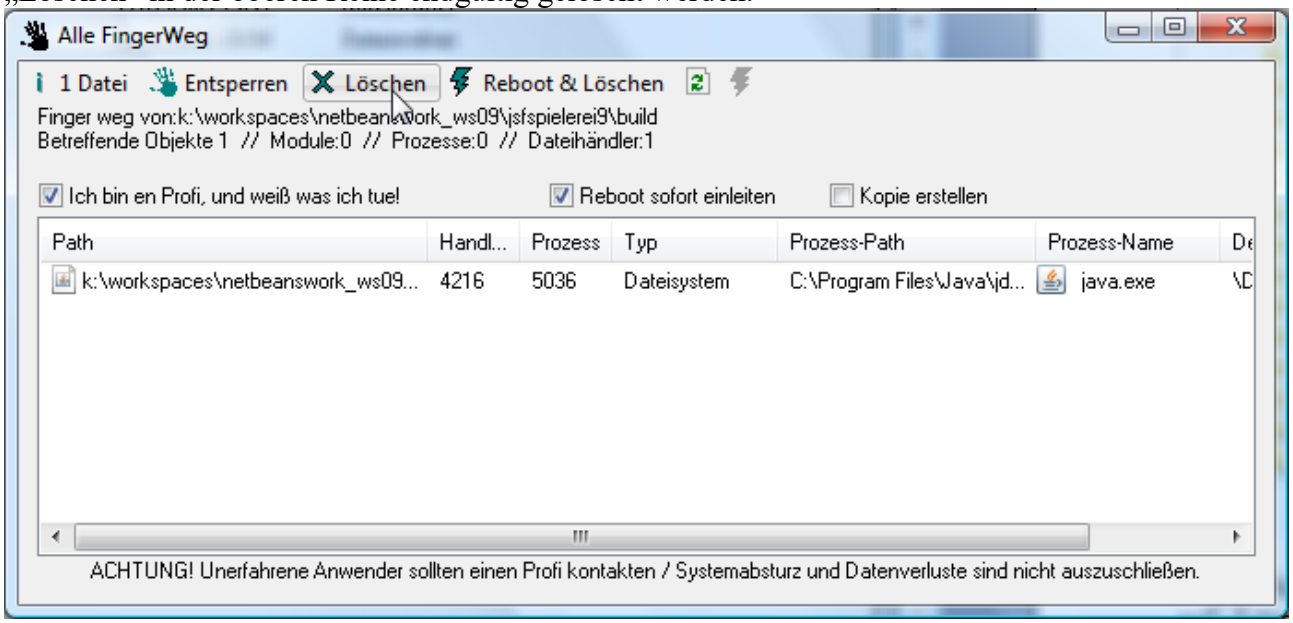
Der Ansatz ist es dann, einfach das build-Verzeichnis des Projekts zu löschen. Hierzu wird das Verzeichnis mit gedrückter linker Maustaste auf das Feld unterhalb von „Datei/Ordner“ gezogen.



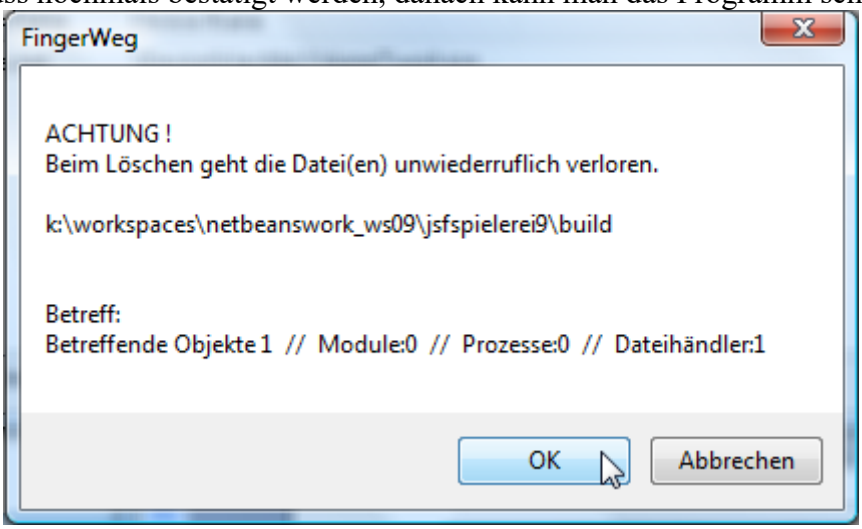
Danach kann der Haken links-unten bei „Kopie erstellen“ gelöscht werden und es wird in der Mitte auf „Löschen“ geklickt.



Es werden dann die nicht löschbaren Dateien gesucht und können dann durch einen Klick auf „Löschen“ in der oberen Reihe endgültig gelöscht werden.



Das Löschen muss nochmals bestätigt werden, danach kann man das Programm schließen.



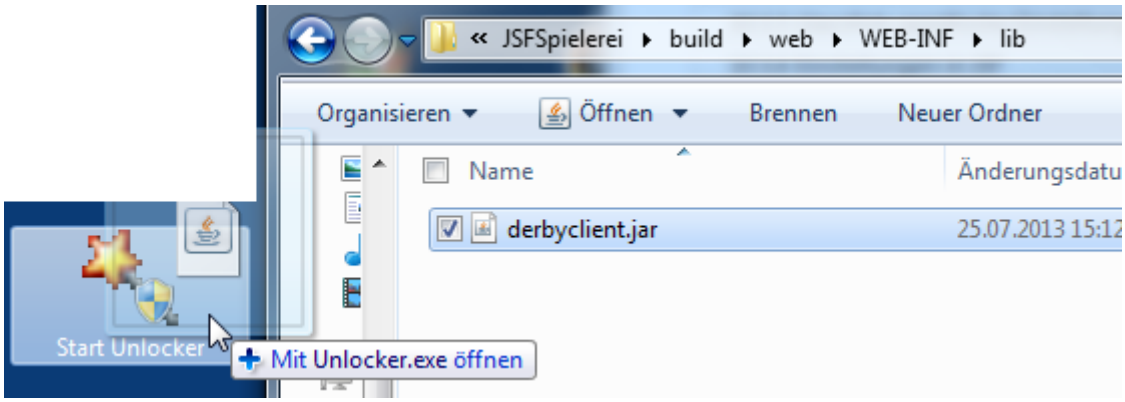
Unlocker

Alternativ ist das Werkzeug „Start Unlocker“ unter http://www.chip.de/downloads/Unlocker_18414122.html vielleicht etwas einfacher nutzbar.

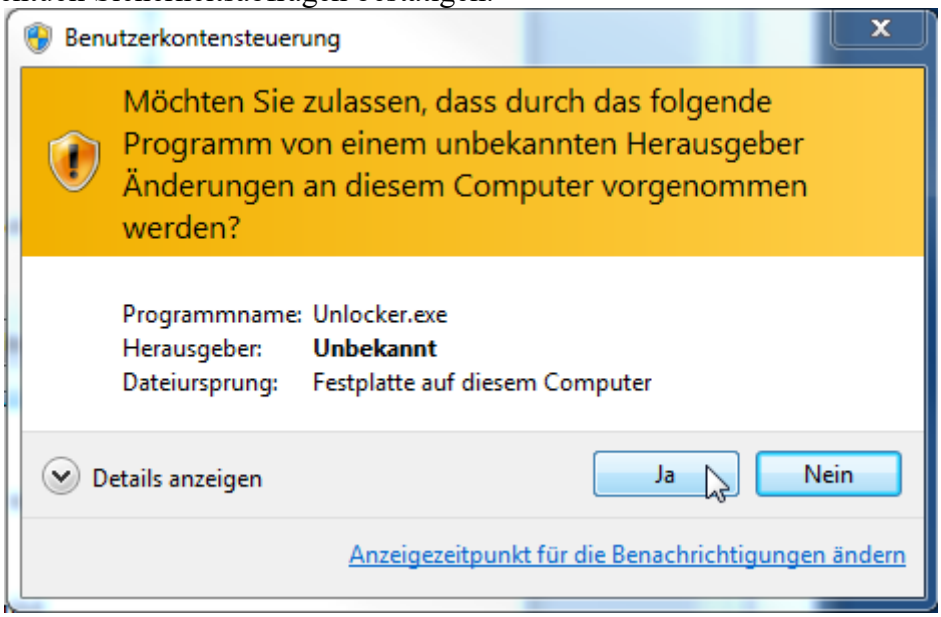
Tritt ein Löschfehler auf, hier:

```
Deleting directory F:\workspaces\NetBeansWork_WS13\JSFSpielerei\build
F:\workspaces\NetBeansWork_WS13\JSFSpielerei\nbproject\build-impl.xml:1396: Unable to delete file
F:\workspaces\NetBeansWork_WS13\JSFSpielerei\build\web\WEB-INF\lib\derbyclient.jar
BUILD FAILED (total time: 1 second)
```

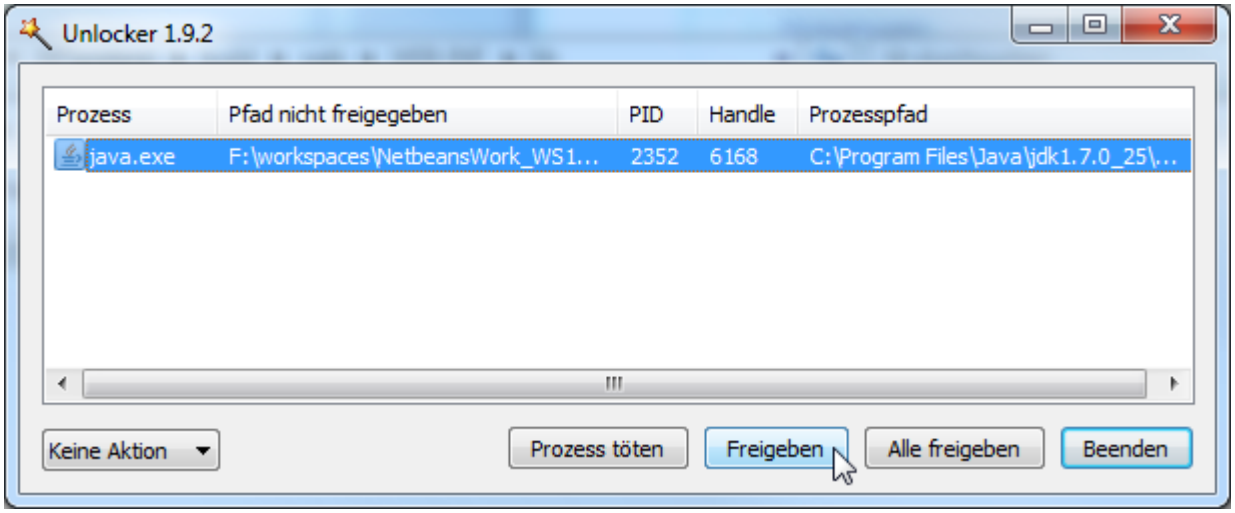
kann die kritische Datei einfach per Drag-and-Drop auf das Icon von „Start Unlocker“ gezogen werden.



Man muss eventuell Sicherheitsabfragen bestätigen.

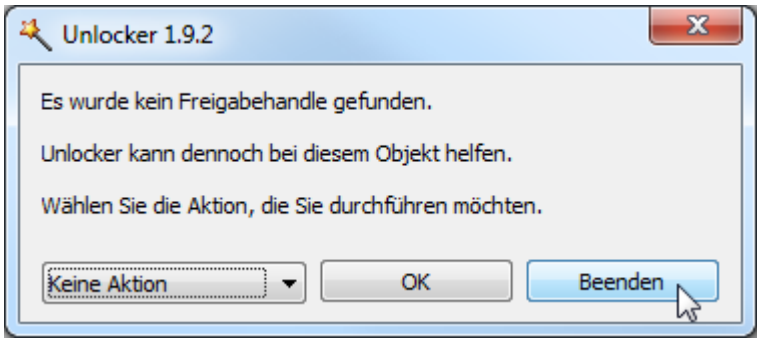


Dann muss man den Prozess java.exe zum „Freigeben“ zwingen. Datei nicht den „Prozess töten“, da man sonst den Server neustarten muss, was generell eine, wenn auch langfristig sehr nervende, gute Idee ist.



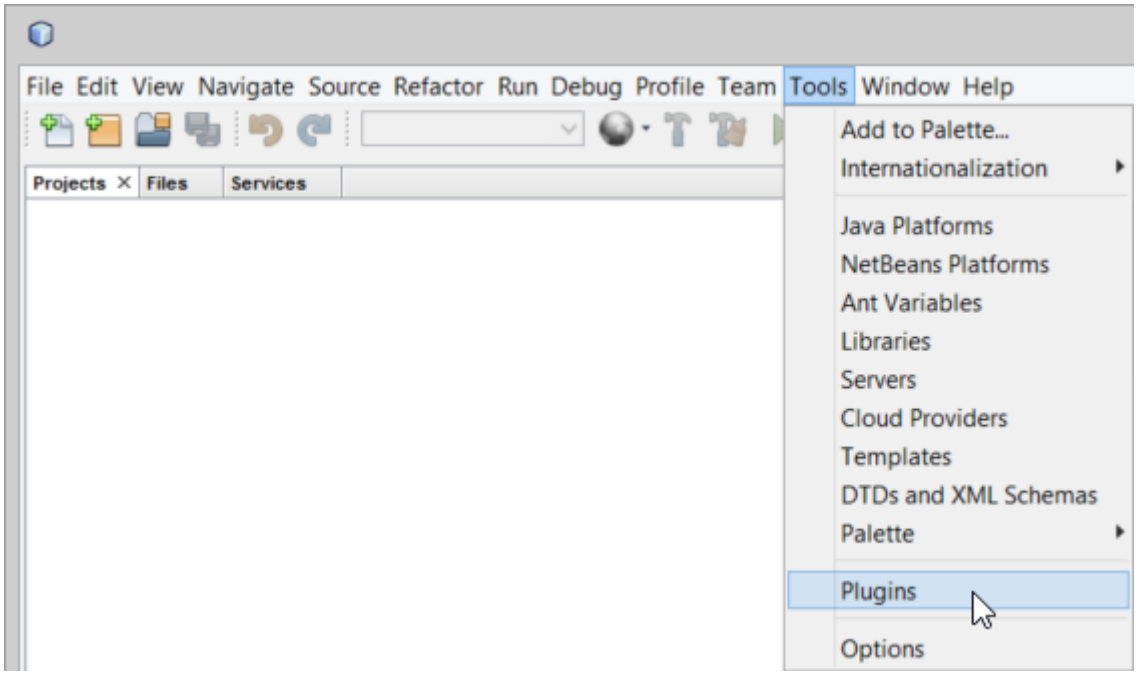
Danach erneut „Build“ durchführen.

Falls die Datei zwischenzeitlich freigegeben wurde, meldet dies Unlocker, dannkann man wieder direkt von NetBeans aus arbeiten.

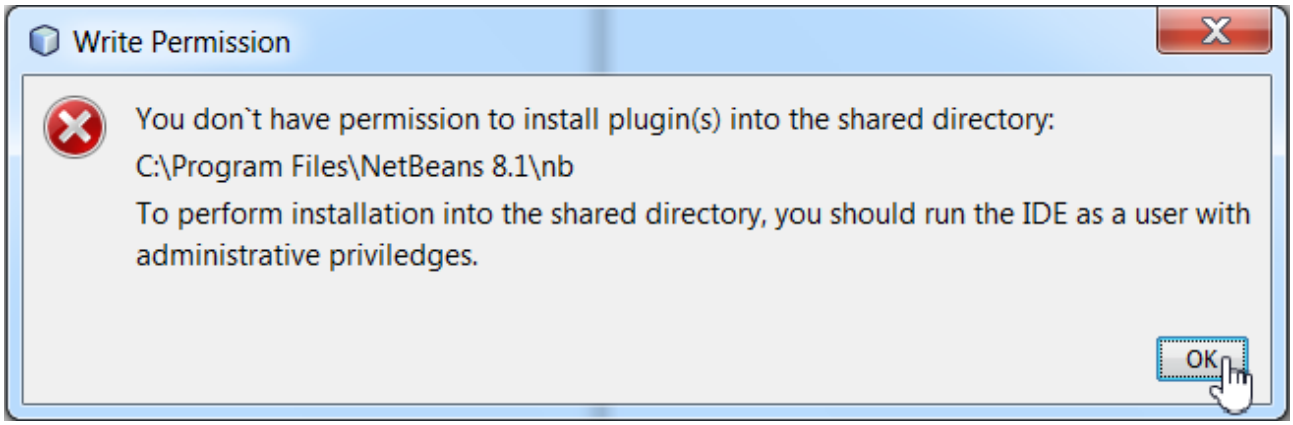


15 Installation von PlugIns - MongoDB Client

Die folgende Installationsbeschreibung kann für beliebige Plugins genutzt werden, die über das Menü „Tools->Plugins“ erreichbar sind. Die genaue Konfiguration des jeweiligen Plugins ist individuell.

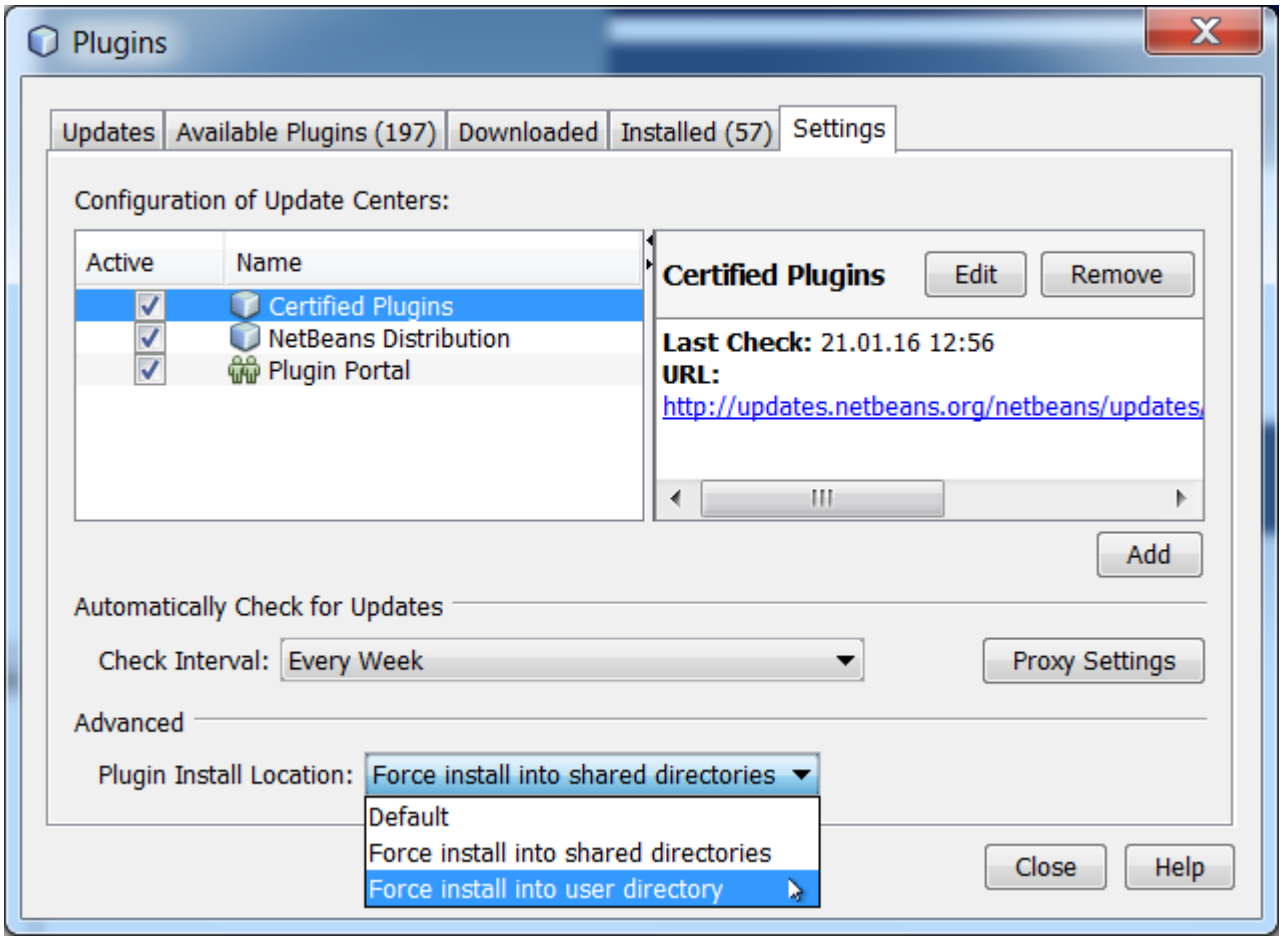


Abhängig vom Rechner, auf dem das Plugin installiert werden soll, kann folgendes Fenster gegebenenfalls erscheinen. Besteht Interesse daran, dass alle Nutzer die Plugins nutzen können, sollte dem Rat NetBeans mit Administratorenrechten zu starten, gefolgt werden.



Die genaue Installationsart wird unter dem Reiter „Settings“ ausgewählt. Sollen die Erweiterungen für alle Nutzer zur Verfügung stehen, ist unter „Plugin Install Location“ die Einstellung „Force Install into shared directions“ auszuwählen, was u. a. dazu führt, dass im NetBeans-Verzeichnis ein

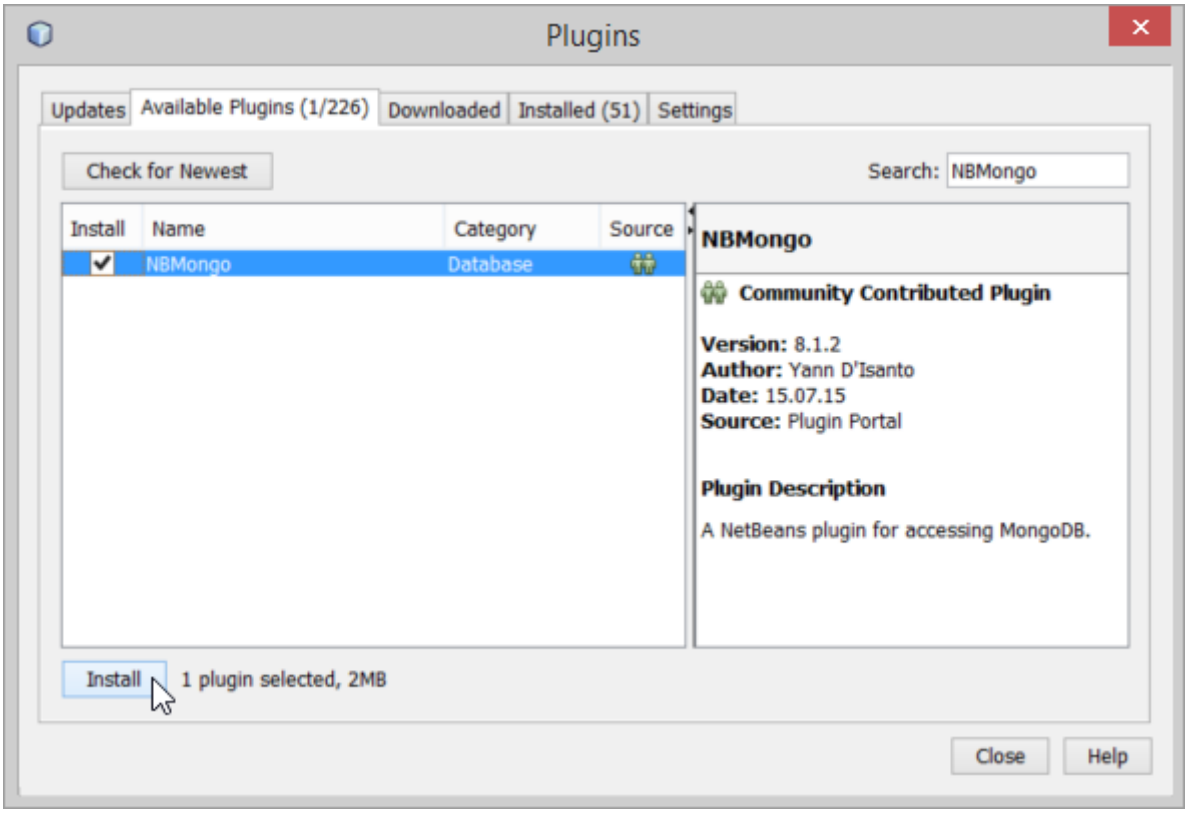
Unterordner „extra“ angelegt wird. Soll die Erweiterung nur für den aktuellen Nutzer durchgeführt werden, da z. B. keine Rechte für ein gemeinsames Verzeichnis vorliegen, wie es z. B. für Studierende an Hochschulrechnern gilt, ist „Force Install into user directory“ auszuwählen.



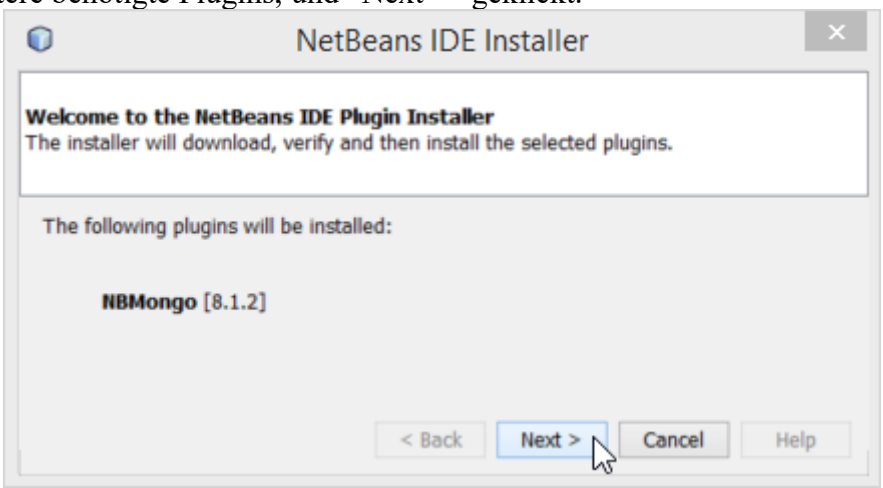
15.1 Installation eines NetBeans-Plugins am Beispiel NBMongo

MongoDB (kurz für: humongous DB, gigantische DB, http://www.mongodb.org/) ist eine dokumentenorientierte Datenbank und stellt damit eine Alternative bzw. Ergänzung zu einer relationalen Datenbank dar. Die Installation und erste Nutzungsschritte sind im Kapitel “17 MongoDB“ beschrieben und werden hier vorausgesetzt. Anders als bei JavaDB ist ein Start der Datenbank aus NetBeans heraus nicht direkt möglich. Dies Kapitel beschreibt die Installation und Nutzung der Erweiterung NBMongo.

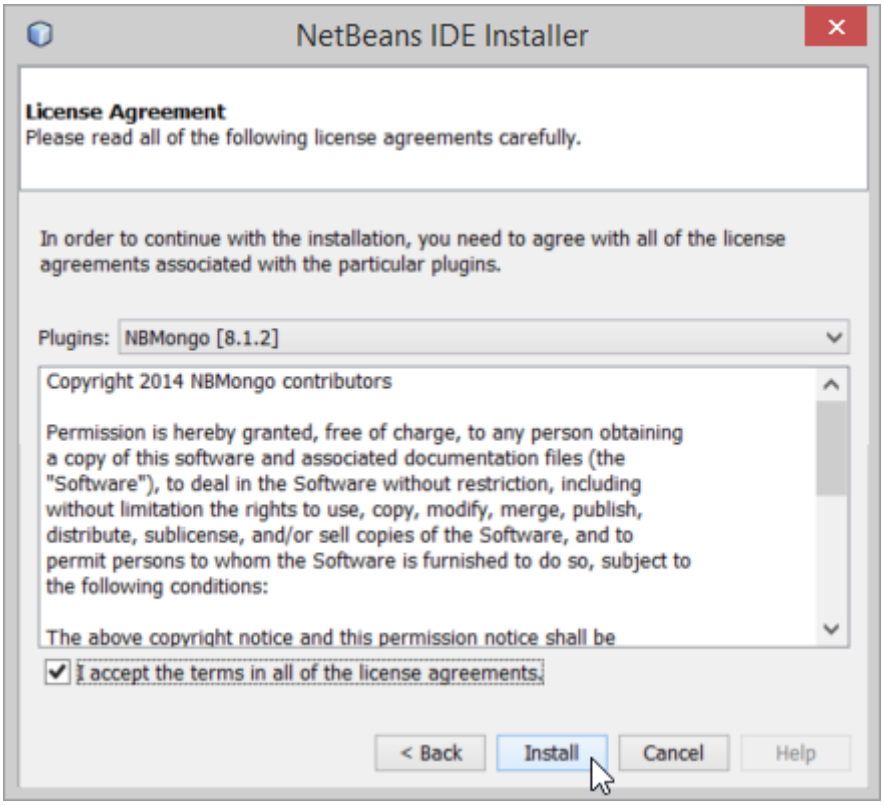
Der Reiter „Available Plugins“ wird angeklickt und rechts-oben nach NBMongo gesucht. Der Haken links-oben in der „Install“-Spalte gesetzt und unten „Install“ geklickt.



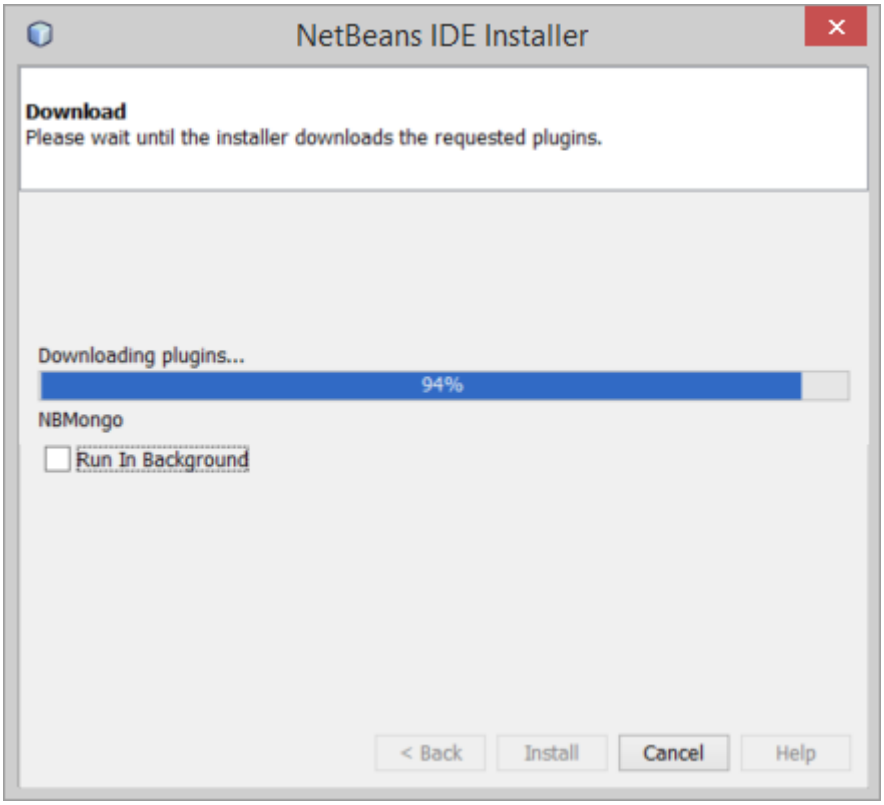
Es wird eine Information über den anstehenden Installationsprozess ausgegeben, gegebenenfalls ergänzt um weitere benötigte Plugins, und “Next >“ geklickt.



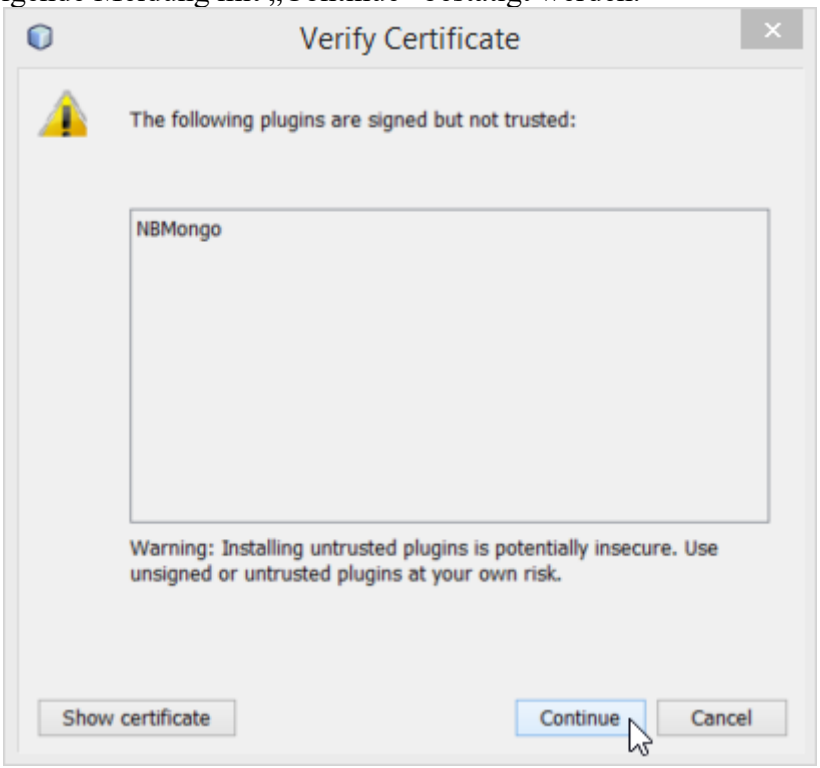
Die Lizenzbedingungen werden gelesen, mit einem Haken links-unten akzeptiert und auf „Install“ geklickt.



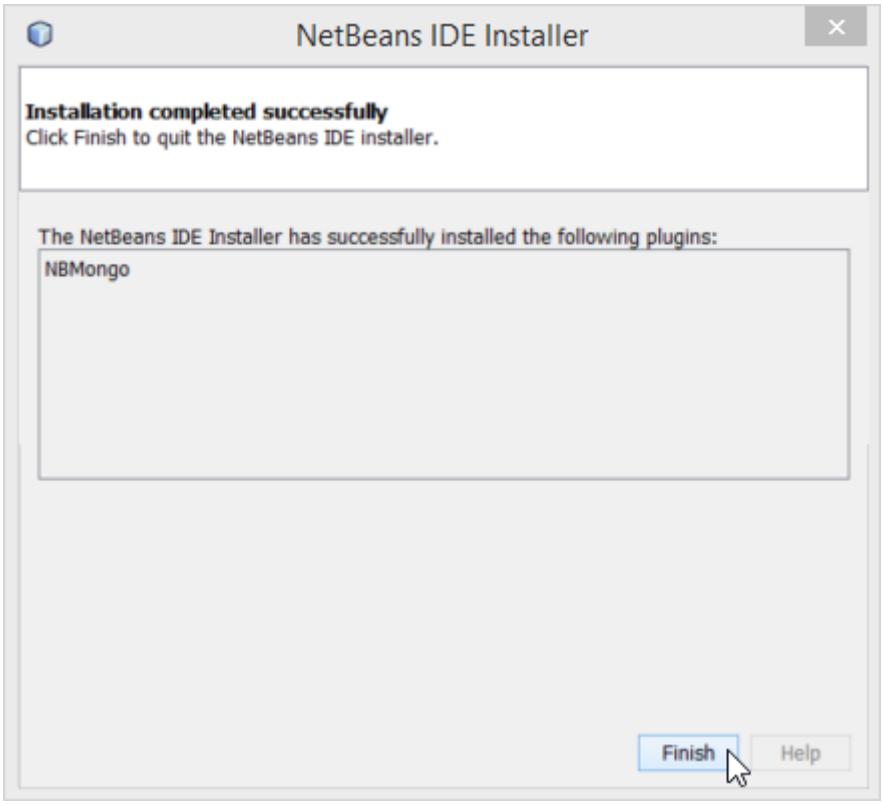
Der Prozess dauert etwas.



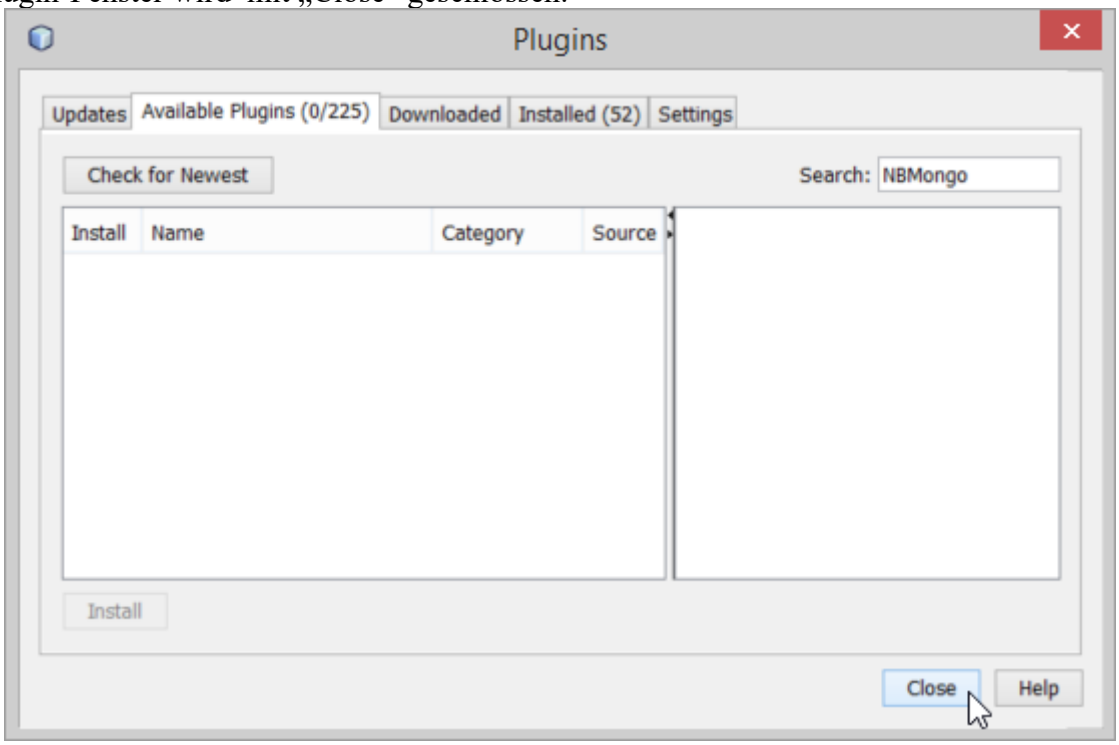
Weiterhin muss folgende Meldung mit „Continue“ bestätigt werden.



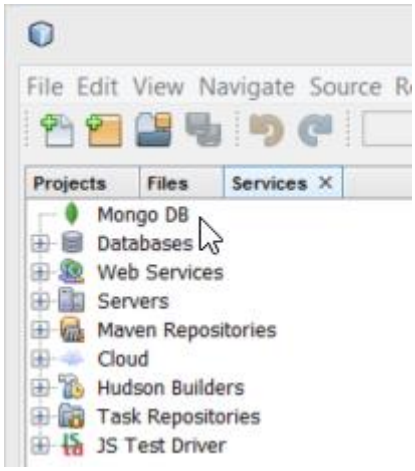
Die erfolgreiche Installation wird bestätigt und mit „Finish“ abgeschlossen. Bei einigen Plugins wird darauf hingewiesen, dass NetBeans neu gestartet werden muss, was dann gemacht werden sollte. Die Installation wird dann beim nächsten Start abgeschlossen.



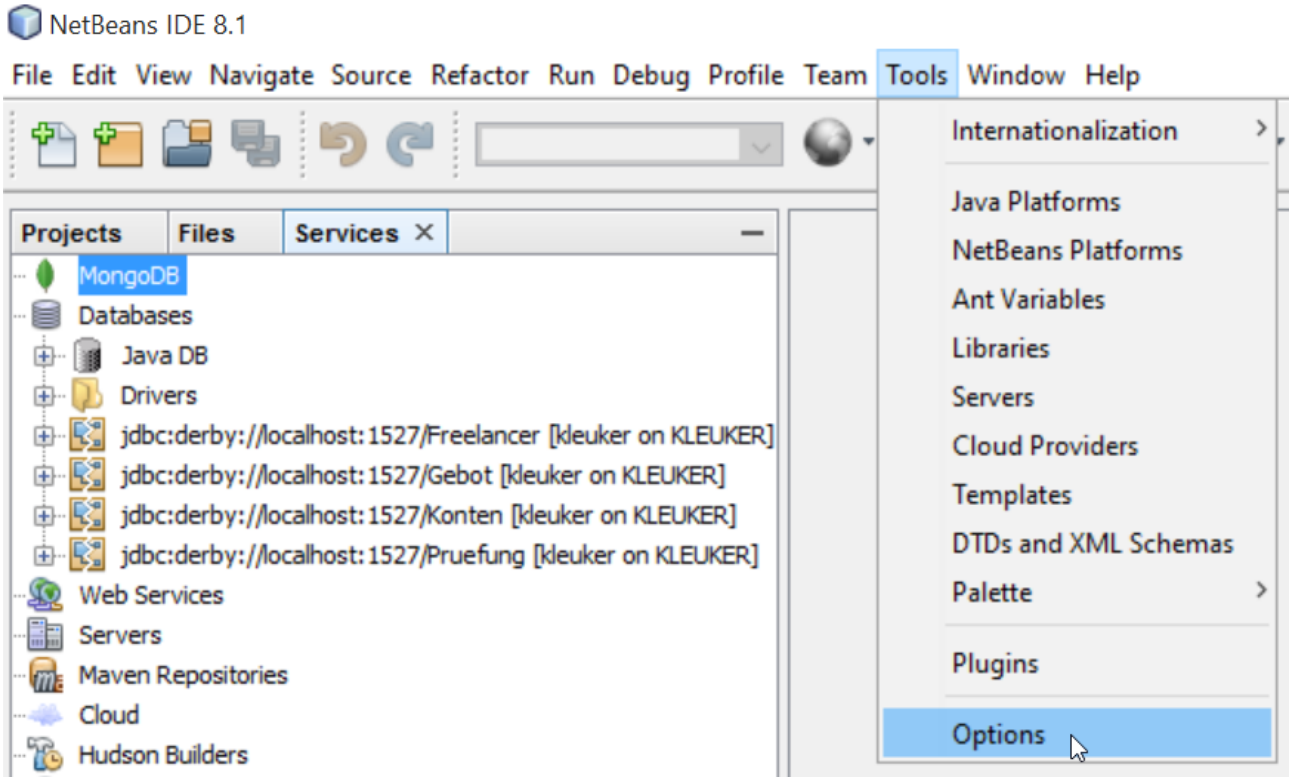
Das Plugin-Fenster wird mit „Close“ geschlossen.



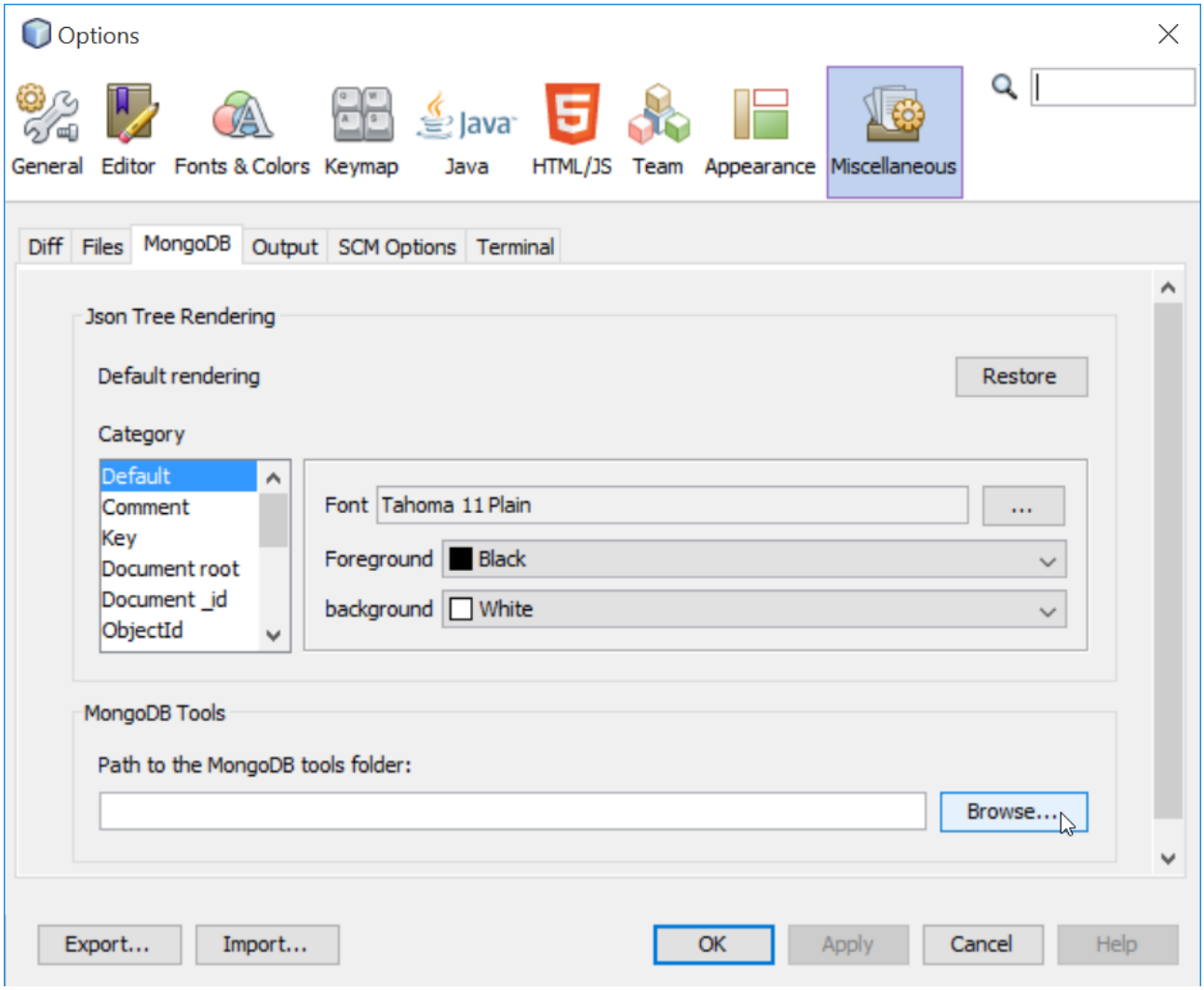
Unter dem Reiter „Services“ befindet sich jetzt ein Eintrag „Mongo DB“.



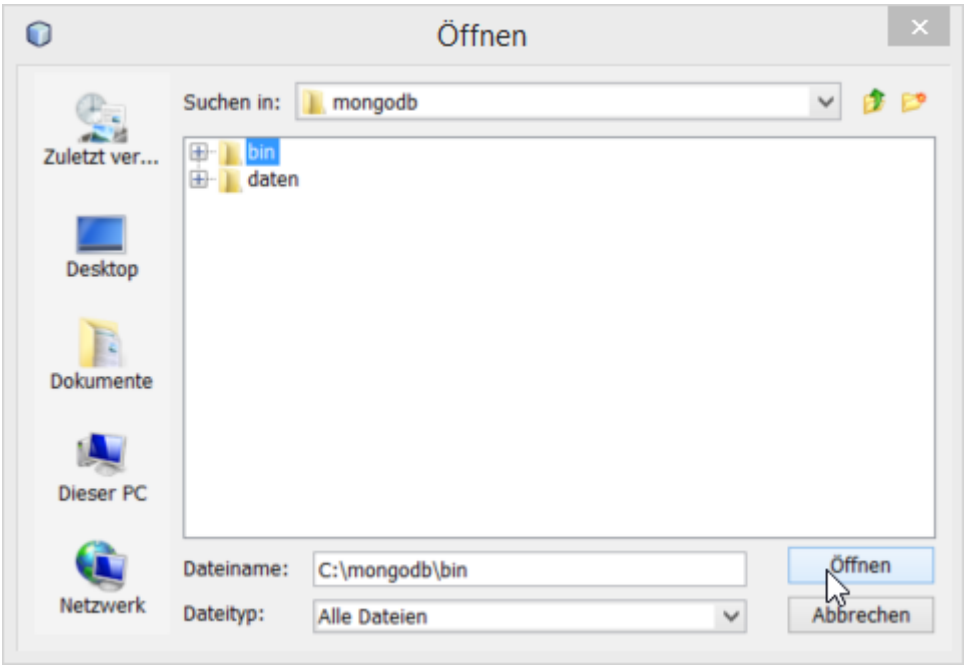
Im nächsten Schritt wird das Installationsverzeichnis der MongoDB in NetBeans eingetragen. Dazu wird ein Klick auf „Tools -> Options“ gemacht.



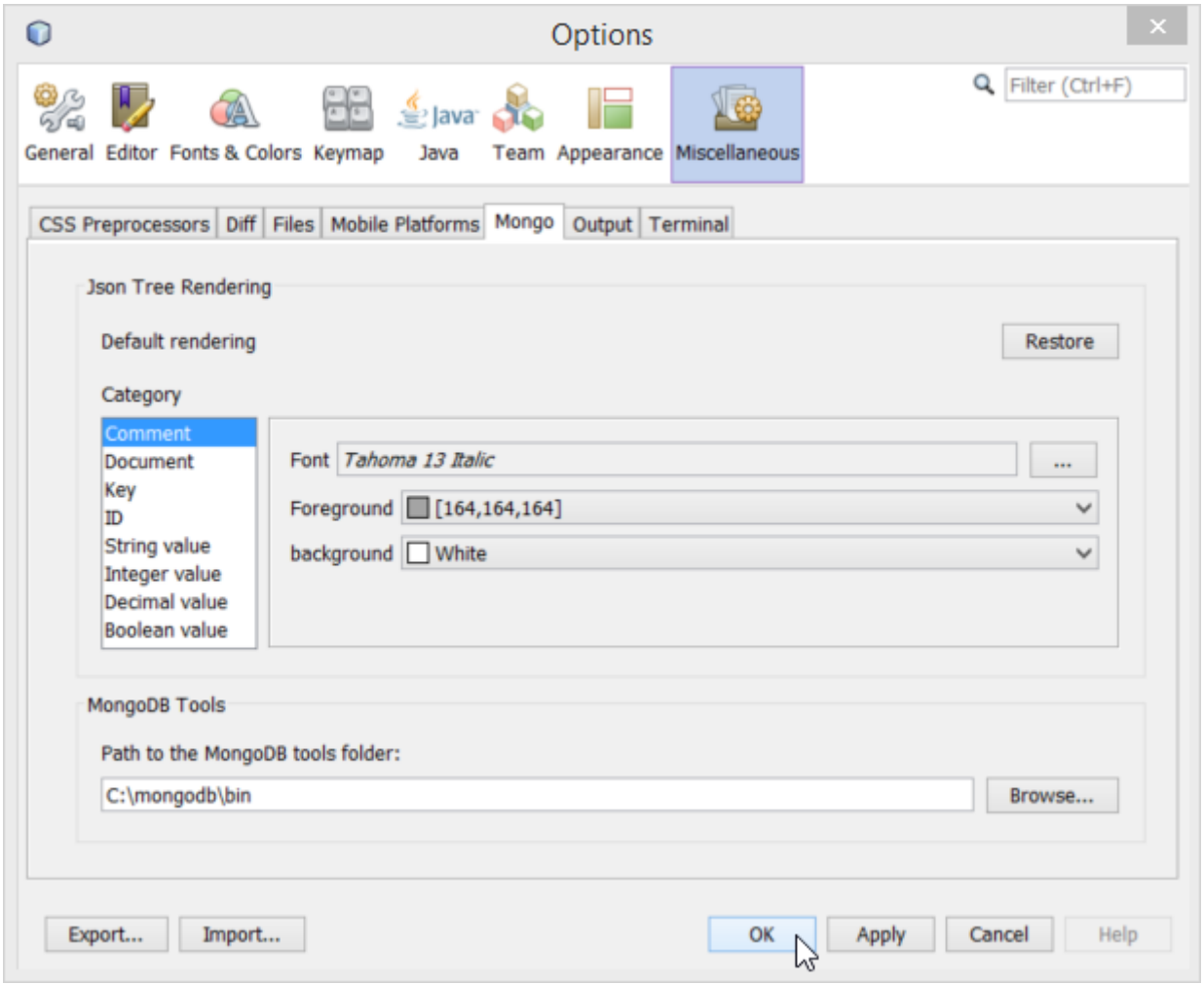
Es wird oben zunächst der Bereich „Miscellaneous“ ausgewählt, auf den Reiter „MongoDB“ und dann rechts-unten auf „Browse…“ geklickt.



Der bin-Ordner der MongoDB-Installation wird angesteuert und auf „Öffnen“ geklickt.

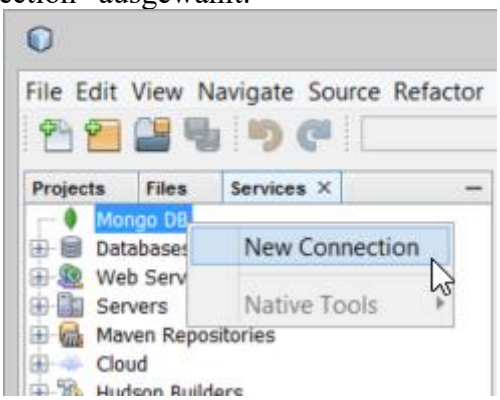


Die Einstellungen werden mit „OK“ abgeschlossen.

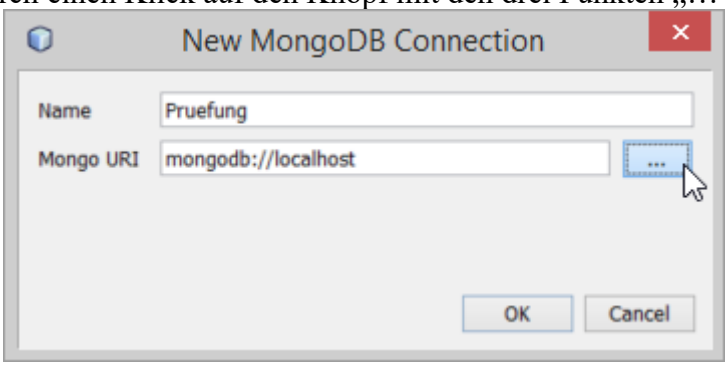


15.2 Aufbau einer Datenbank-Verbindung

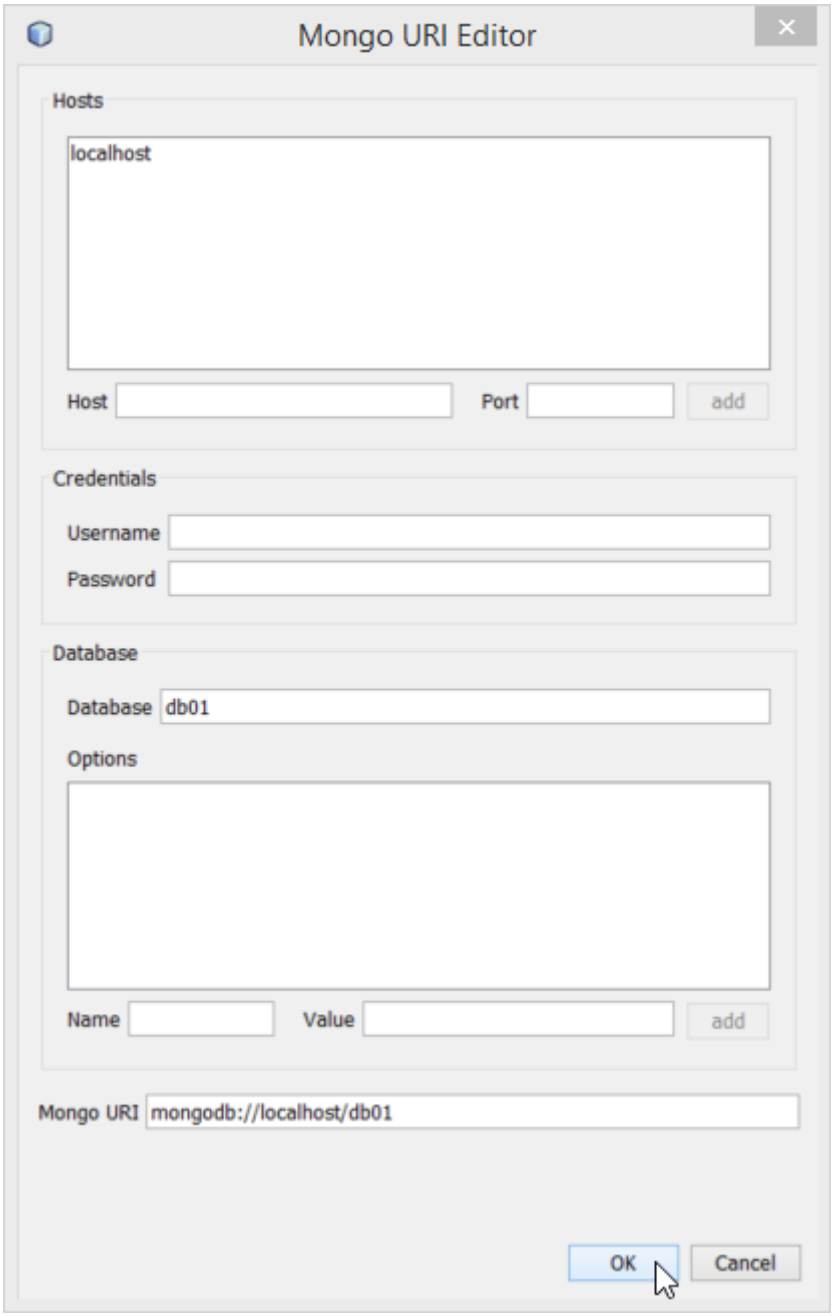
Nachdem die MongoDB außerhalb von NetBeans gestartet wurde, wird ein Rechtsklick auf „Mongo DB“ gemacht und „New Connection“ ausgewählt.



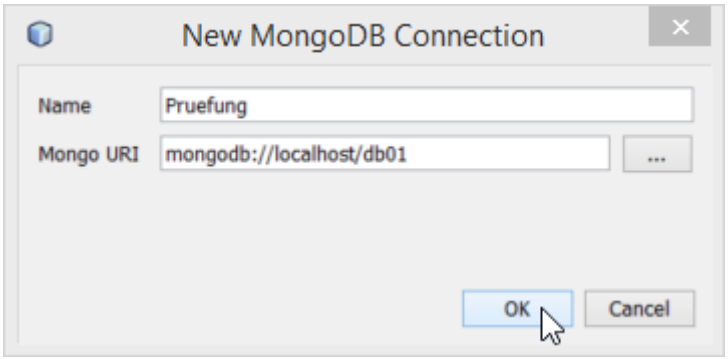
Im oberen Feld „Name“ kann der Verbindung ein beliebiger Name gegeben werden. Detailliertere Einstellungen sind durch einen Klick auf den Knopf mit den drei Punkten „…“ möglich.



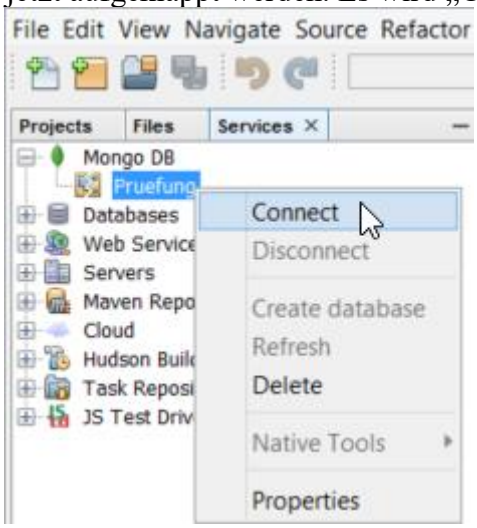
Generell müssen jetzt die Verbindungsinformationen eingetragen werden. Hat man die einfachsten Installationseinstellungen für die MongoDB genutzt, reicht es aus, den Datenbanknamen unter „Database“, hier „db01“, einzutragen und auf „OK“ zu klicken.



Die Einrichtung wird mit „OK“ abgeschlossen.

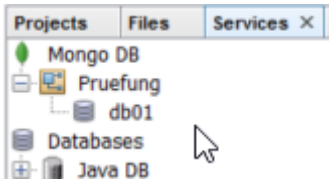


Der Eintrag „Mongo DB“ kann jetzt aufgeklappt werden. Es wird „Connect“ ausgewählt.



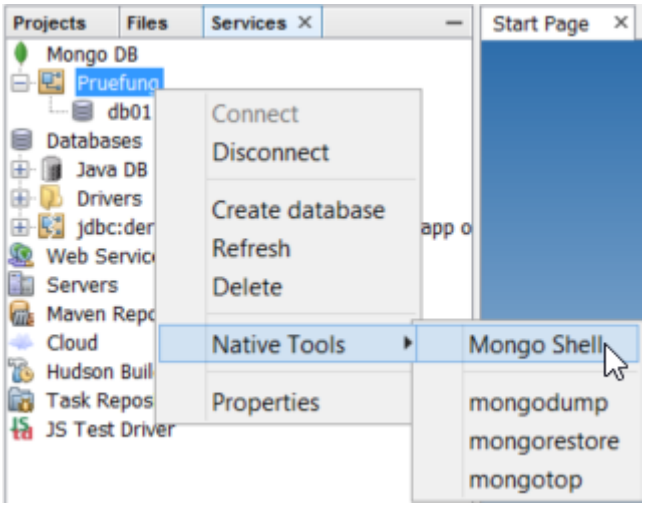
Sollte die Datenbank nicht laufen, erhält man eine nicht ganz klare Fehlermeldung oder nur die Information „Please Wait…“.

Wird die Verbindung erfolgreich hergestellt, kann man den Datenbankeintrag aufklappen und sieht die bereits vorhandenen Collections oder die noch leere Datenbank.

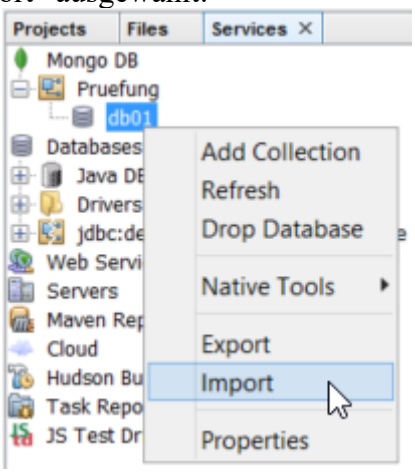


15.3 Erste Nutzung des MongoDB-Clients

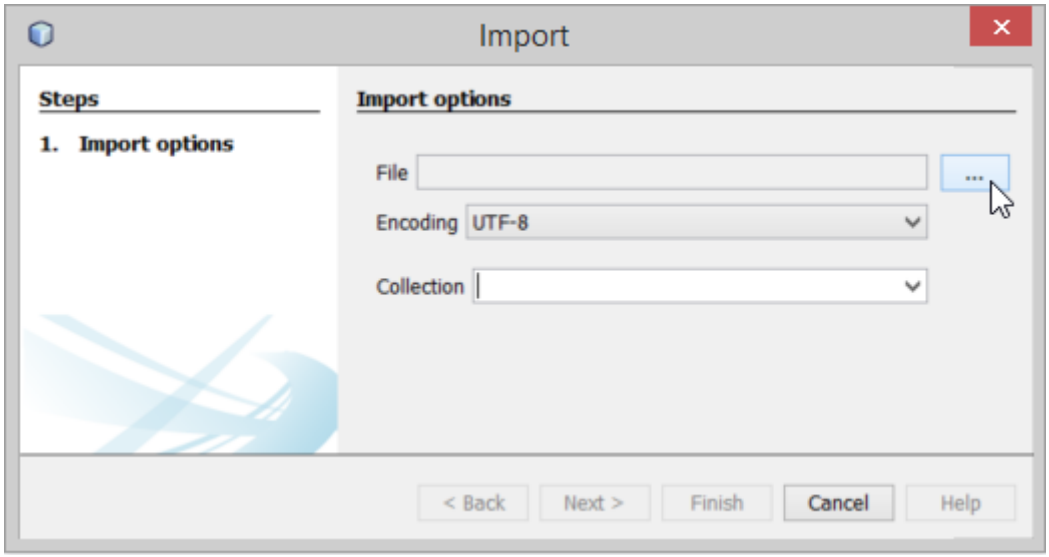
Der Client bietet einige Zugriffe auf die Daten, die die Arbeit mit der MongoDB etwas erleichtern. Hat man sich bereits mit der Konsolenarbeit mit MongoDB und JavaScript beschäftigt, steht diese Möglichkeit auch über einen Rechtsklick auf der Datenbank unter „Native Tools -> Mongo Shell“ zur Verfügung.



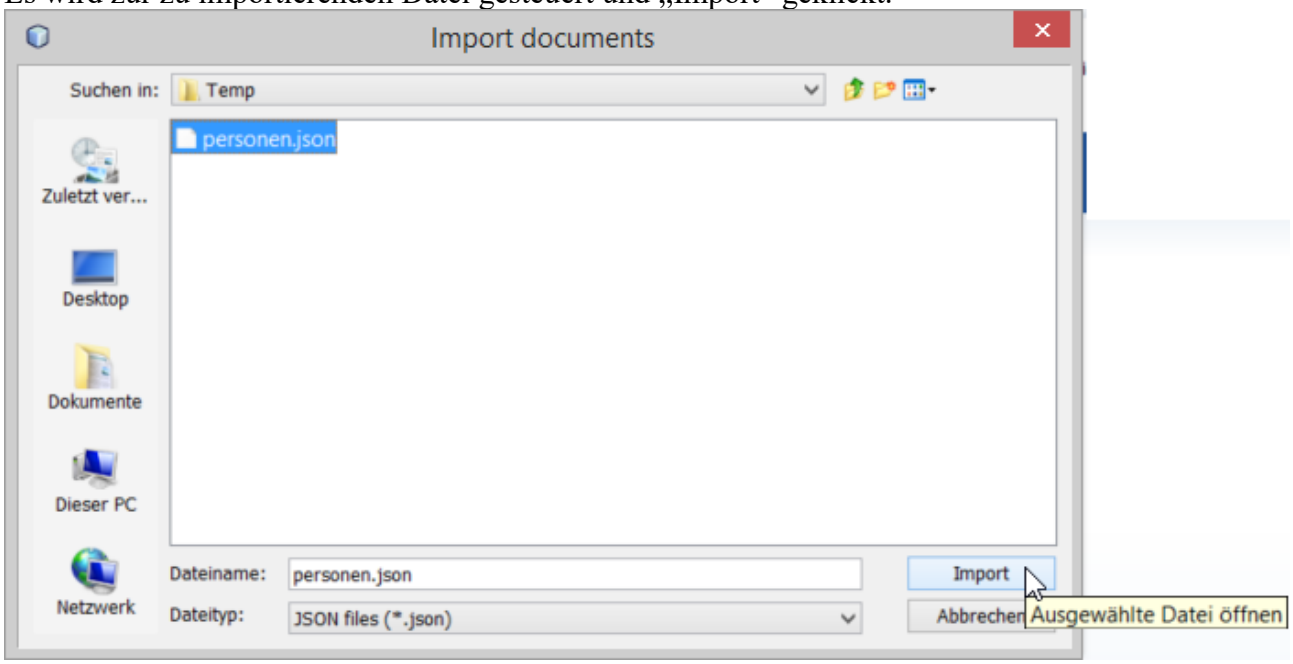
Der Client unterstützt das Importieren von Daten im JSON-Format. Dazu wird ein Rechtsklick auf der Datenbank gemacht und „Import“ ausgewählt.



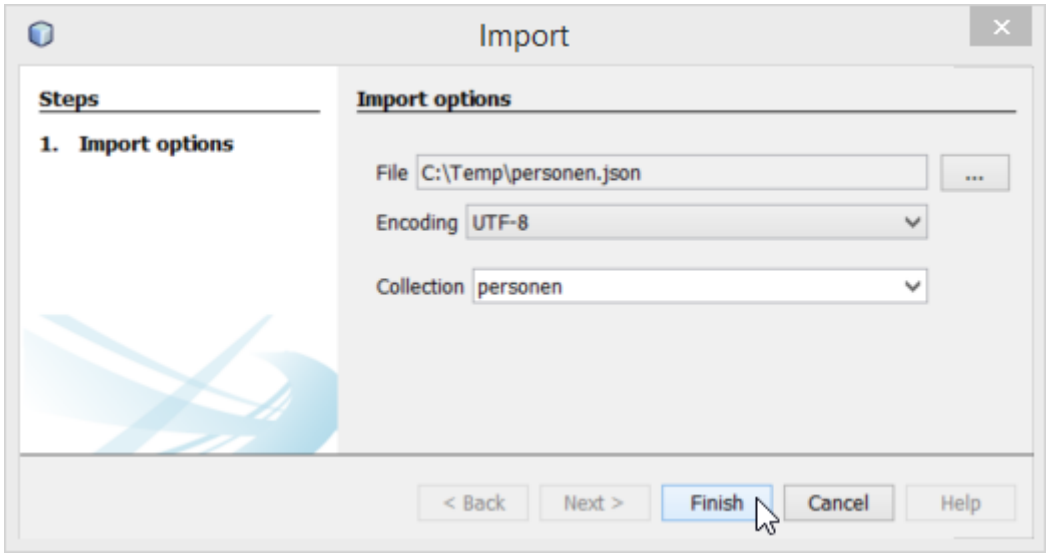
Oben-rechts wird auf die drei Punkte geklickt.



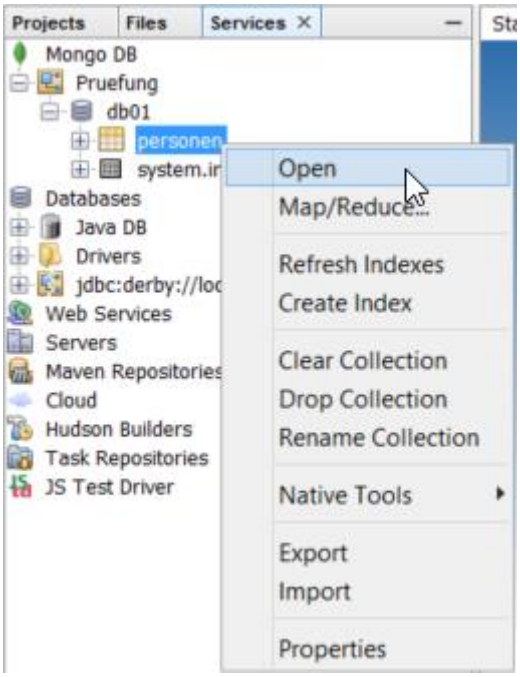
Es wird zur zu importierenden Datei gesteuert und „Import“ geklickt.



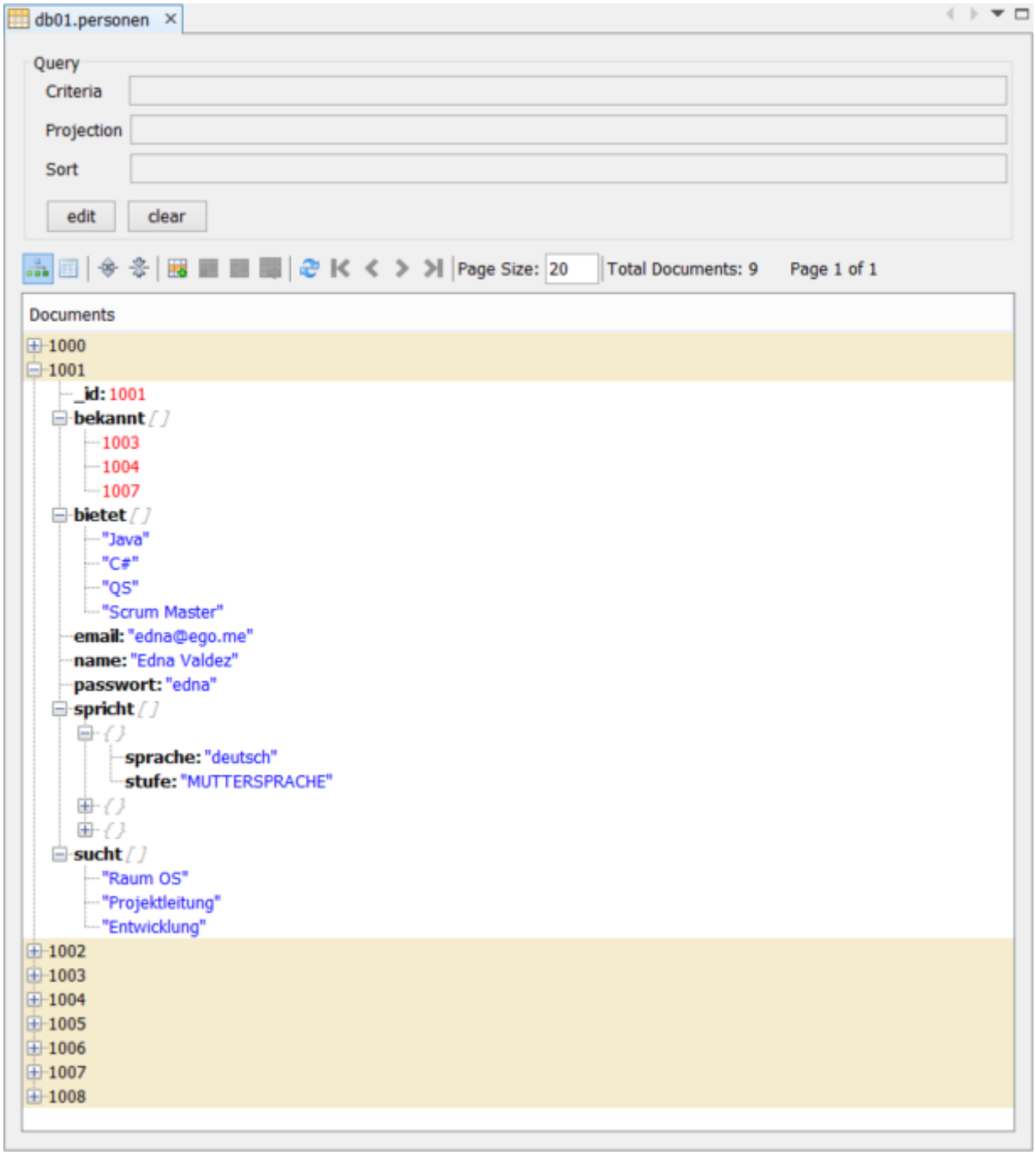
Weiter unten kann man entweder eine existierende Collection auswählen oder den Namen einer neuen Collection eingeben. Abschließend wird auf „Finish“ geklickt.



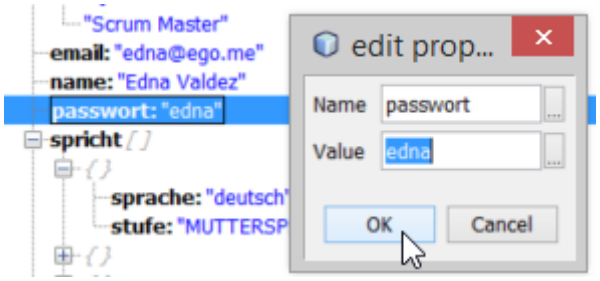
Um mit den Daten im Client zu arbeiten, wird auf der relevanten Collection ein Rechtsklick gemacht und „Open“ ausgewählt.



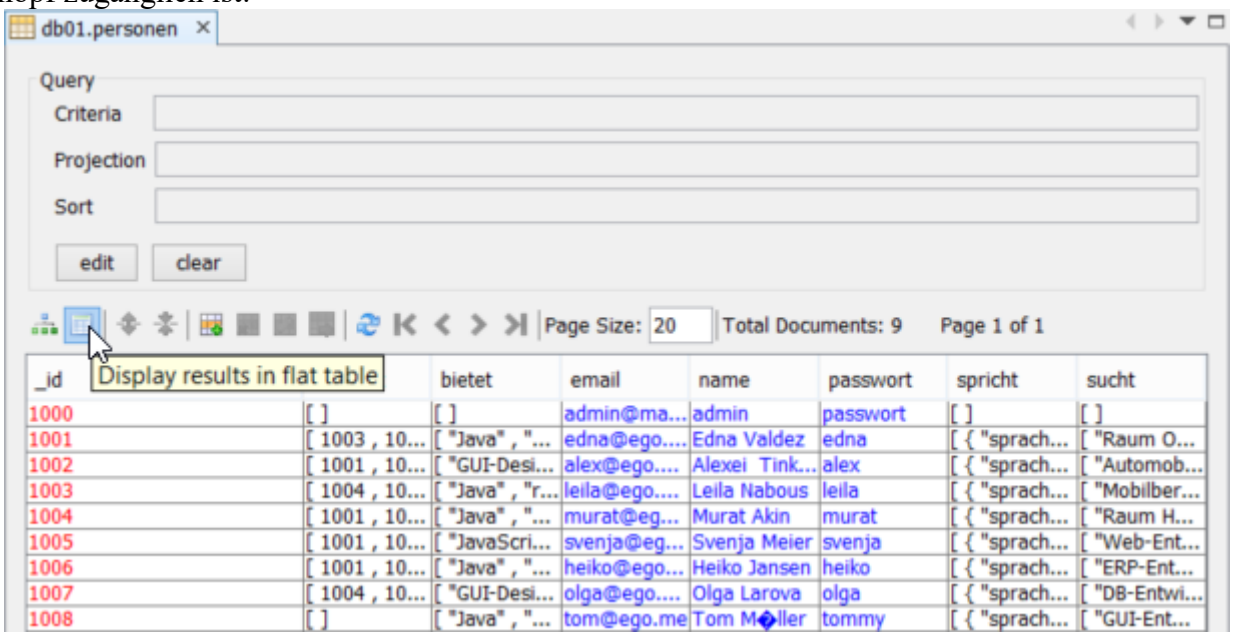
In der unteren Hälfte werden alle Dokumente angezeigt, die zum Ergebnis einer Anfrage gehören. Nach dem Öffnen werden alle Dokumente angezeigt. Diese können aufgeklappt und analysiert werden.



Wie in anderen Ansichten auch, kann man auf einzelne Attribut-Wert-Paare klicken, sich den Inhalt anschauen und bearbeiten.

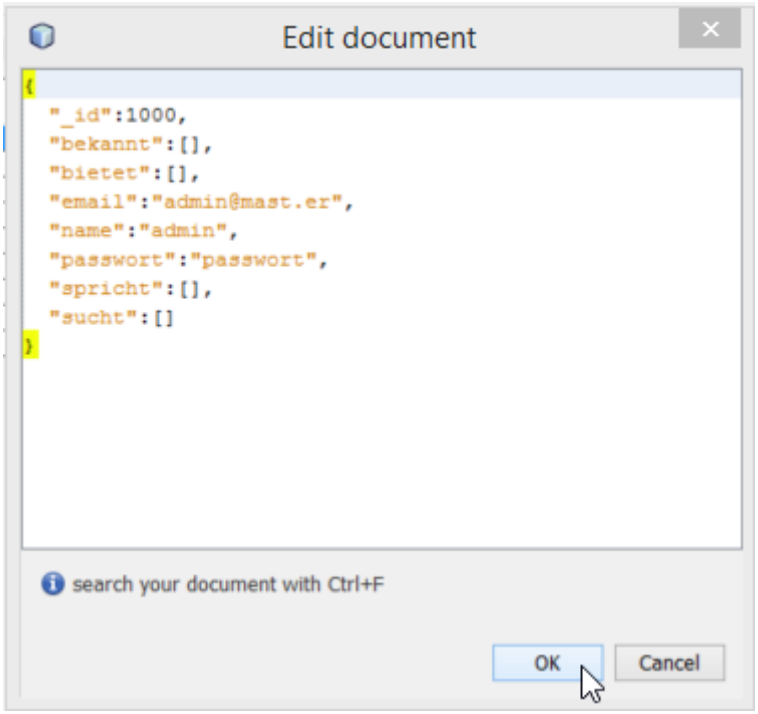


Wenn die Daten eher linear sind, kann auch eine Tabellenansicht hilfreich sein, die unter dem zweiten Knopf zugänglich ist.

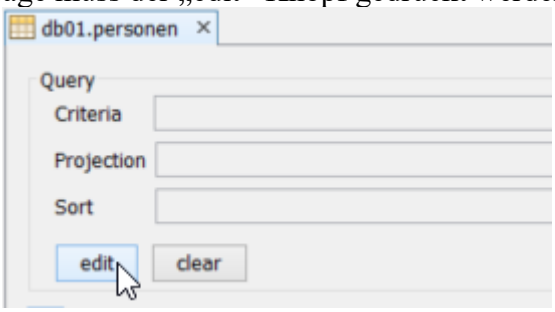


_id		bietet	email	name	passwort	spricht	sucht
1000	[ ]	[ ]	admin@ma...	admin	passwort	[ ]	[ ]
1001	[1003 , 10...	["Java" , "...	edna@ego....	Edna Valdez	edna	[{ "sprach...	["Raum O...
1002	[1001 , 10...	["GUI-Desi...	alex@ego....	Alexei Tink...	alex	[{ "sprach...	["Automob...
1003	[1004 , 10...	["Java" , "r...	leila@ego....	Leila Nabous	leila	[{ "sprach...	["Mobilber...
1004	[1001 , 10...	["Java" , "...	murat@eg...	Murat Akin	murat	[{ "sprach...	["Raum H...
1005	[1001 , 10...	["JavaScri...	svenja@eg...	Svenja Meier	svenja	[{ "sprach...	["Web-Ent...
1006	[1001 , 10...	["Java" , "...	heiko@ego...	Heiko Jansen	heiko	[{ "sprach...	["ERP-Ent...
1007	[1004 , 10...	["GUI-Desi...	olga@ego....	Olga Larova	olga	[{ "sprach...	["DB-Entwi...
1008	[ ]	["Java" , "...	tom@ego.me	Tom M□ller	tommy	[{ "sprach...	["GUI-Ent...

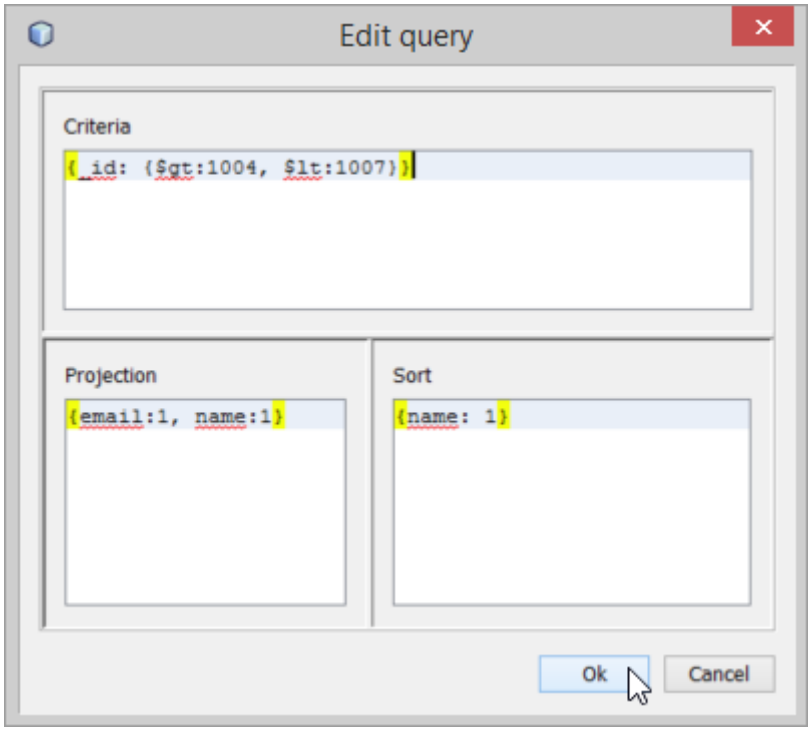
Durch einen Doppelklick auf einem Dokument wird dieses im Detail angezeigt und kann auch bearbeitet werden.



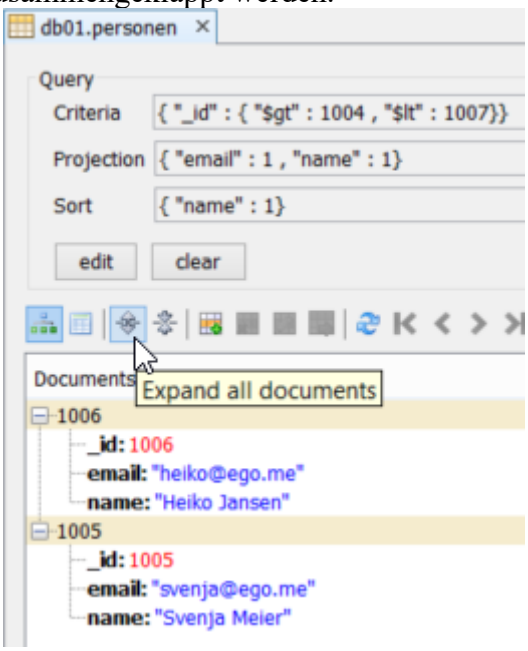
Zur Formulierung einer Anfrage muss der „edit“-Knopf gedrückt werden.



Es können dann die gleichen Kriterien, wie in der JavaScript-Konsole in dem Eingabefenster genutzt werden. Die Fehlermeldungen sind meist wenig aussagekräftig.

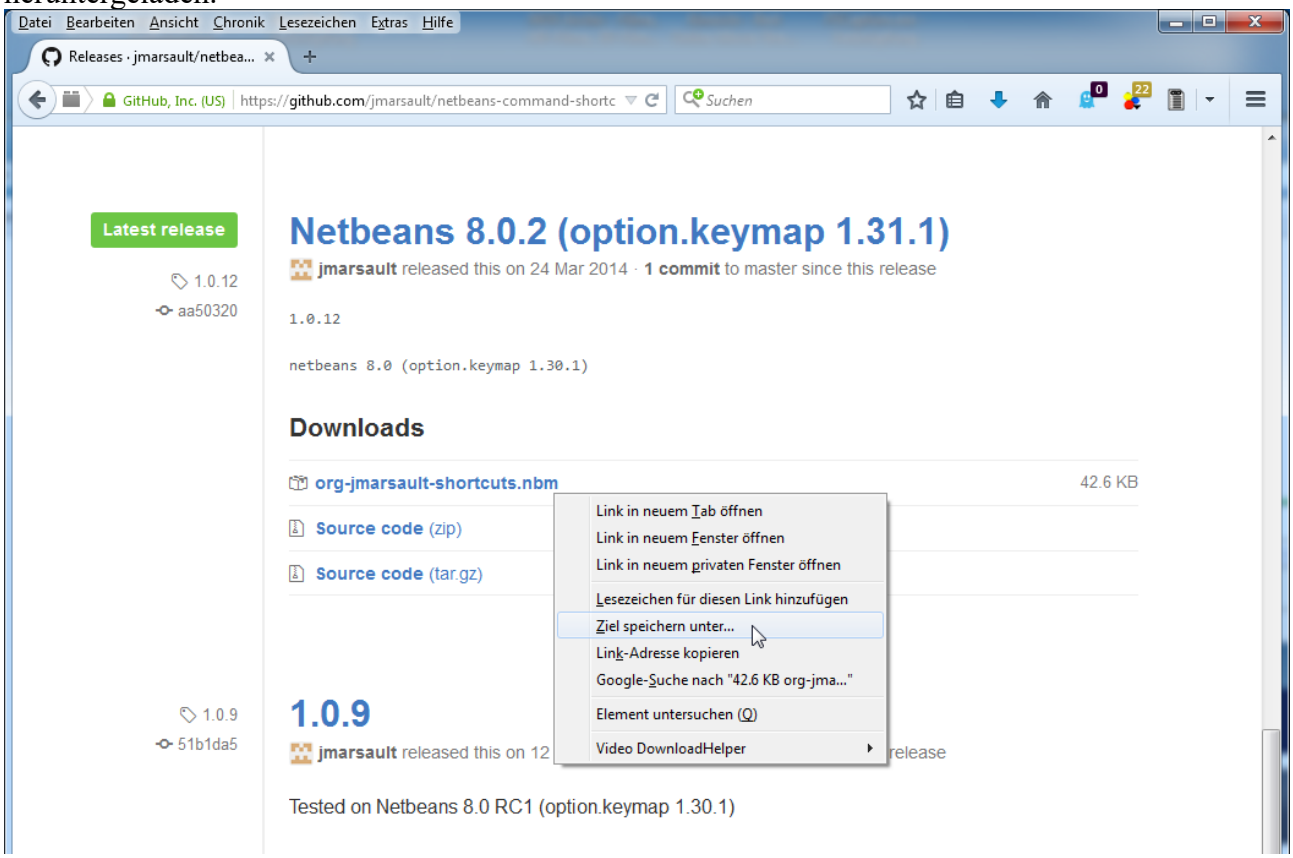


Das Ergebnis wird in der ausgewählten Form angezeigt. Mit „Edit“ können neue Anfragen, mit „Clear“ die Ergebnisse gelöscht werden. Mit dem dritten Knopf können alle Elemente ausgeklappt und mit dem vierten Knopf zusammengeklappt werden.

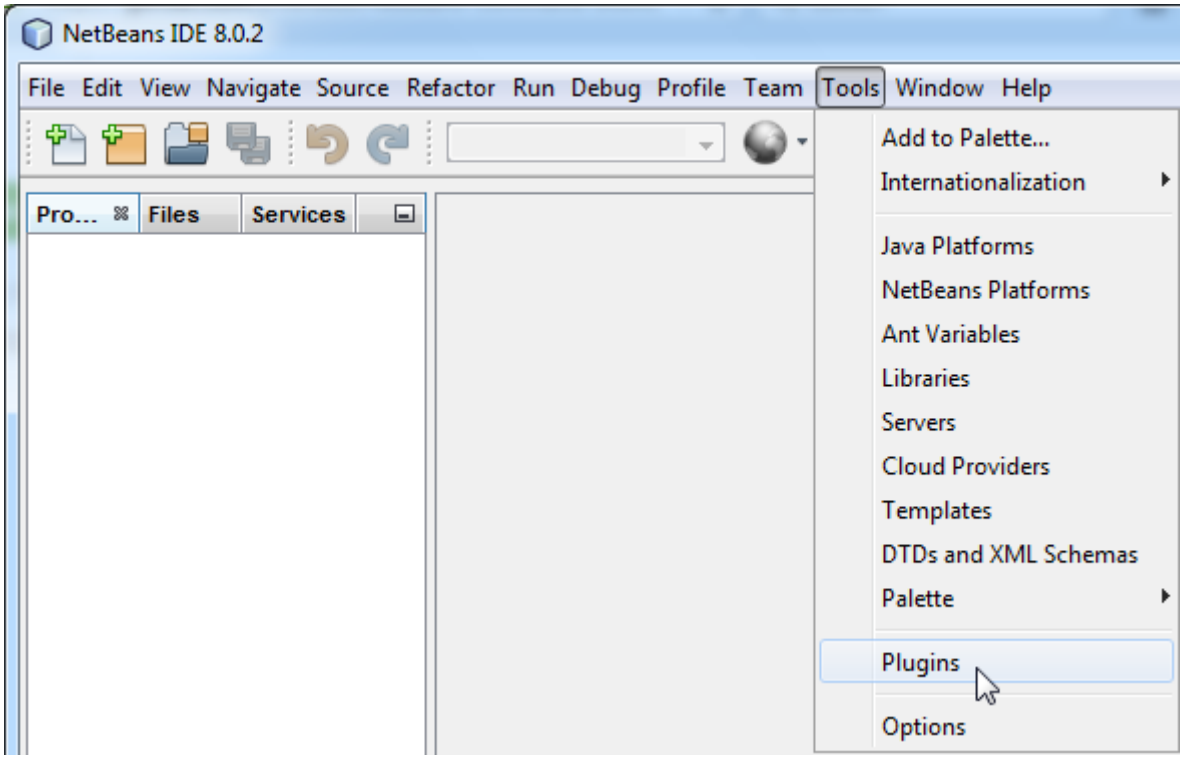


15.4 Installation eines heruntergeladenen Plugins - NetBeans Command Shortcuts

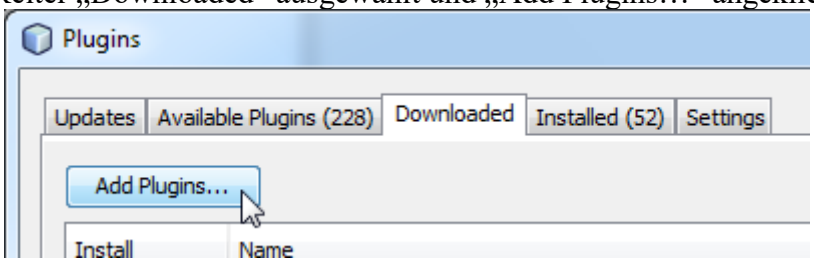
Nicht alle Erweiterungen sind direkt unter NetBeans erreichbar und müssen dann manuell installiert werden, wobei der zentrale spätere Ablauf identisch ist. Als Beispiel wird das Plugin „NetBeans Command Shortcuts" genutzt, mit dem fast beliebige Befehle aus NetBeans heraus startbar sind. Das Plugin ist von der Seite https://github.com/jmarsault/netbeans-command-shortcuts/releases erhältlich. NetBeans-Erweiterungen haben die Endung „.nbm", es wird die passende Version heruntergeladen.



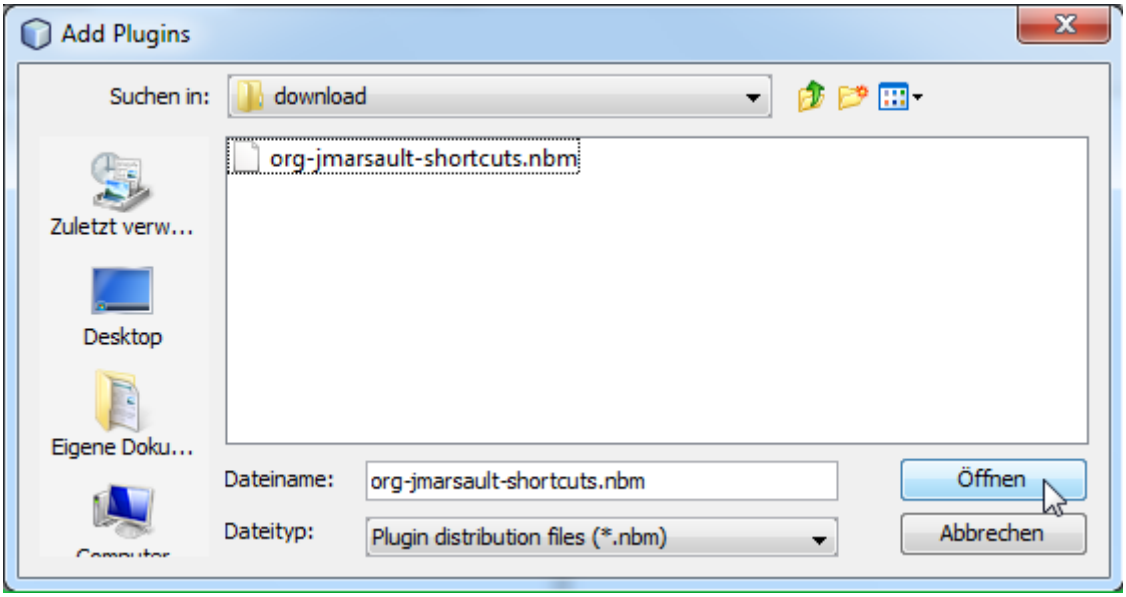
Die eigentliche Installation nutzt wieder den Plugin-Mechanismus, der über „Tools ->Plugins" aufgerufen wird.



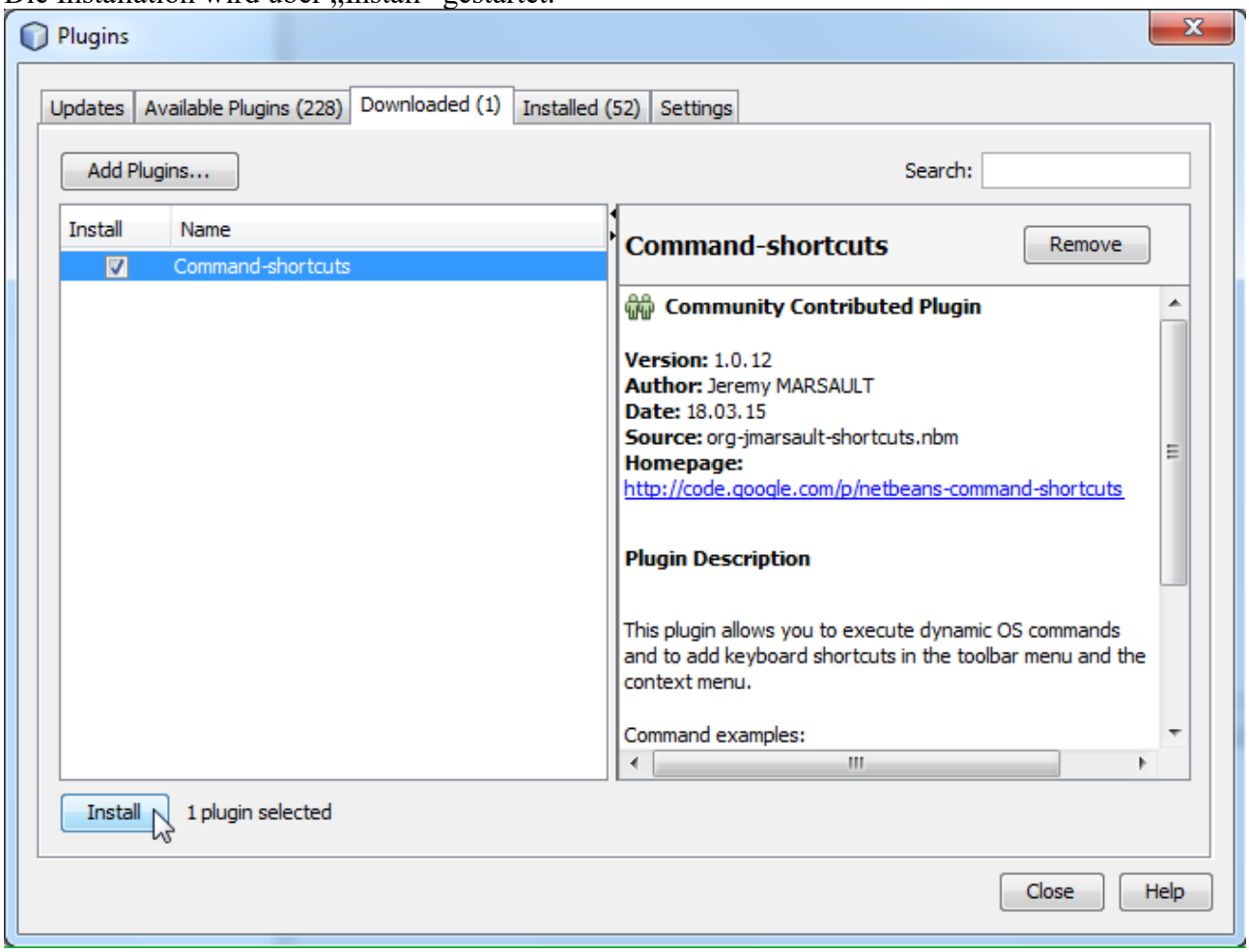
Es wird oben der Reiter „Downloaded“ ausgewählt und „Add Plugins…“ angeklickt.



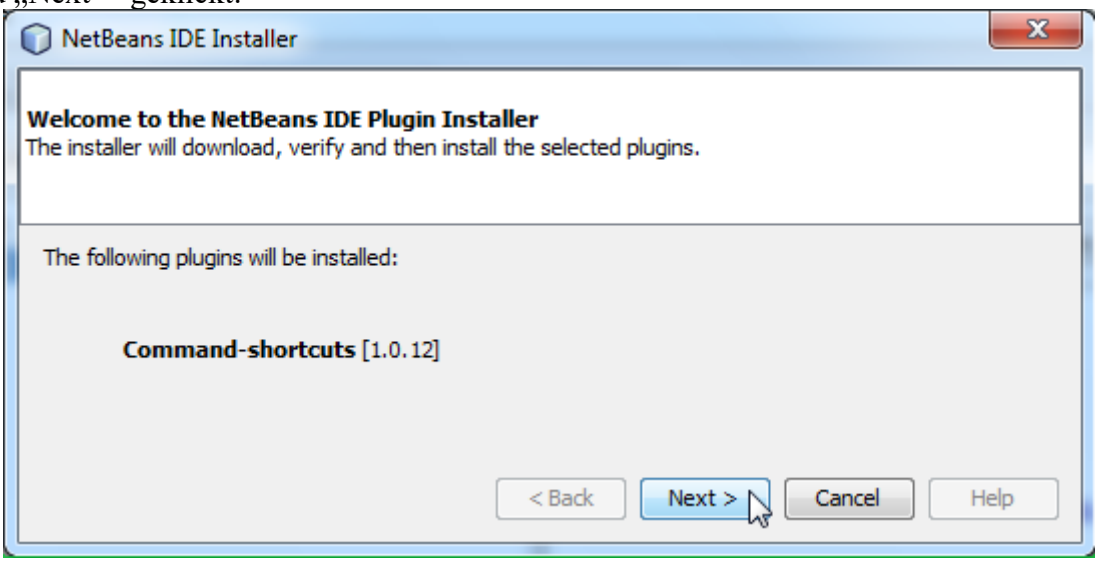
Es wird die heruntergeladene Datei gesucht und mit „Öffnen“ geladen.



Die Installation wird über „Install“ gestartet.



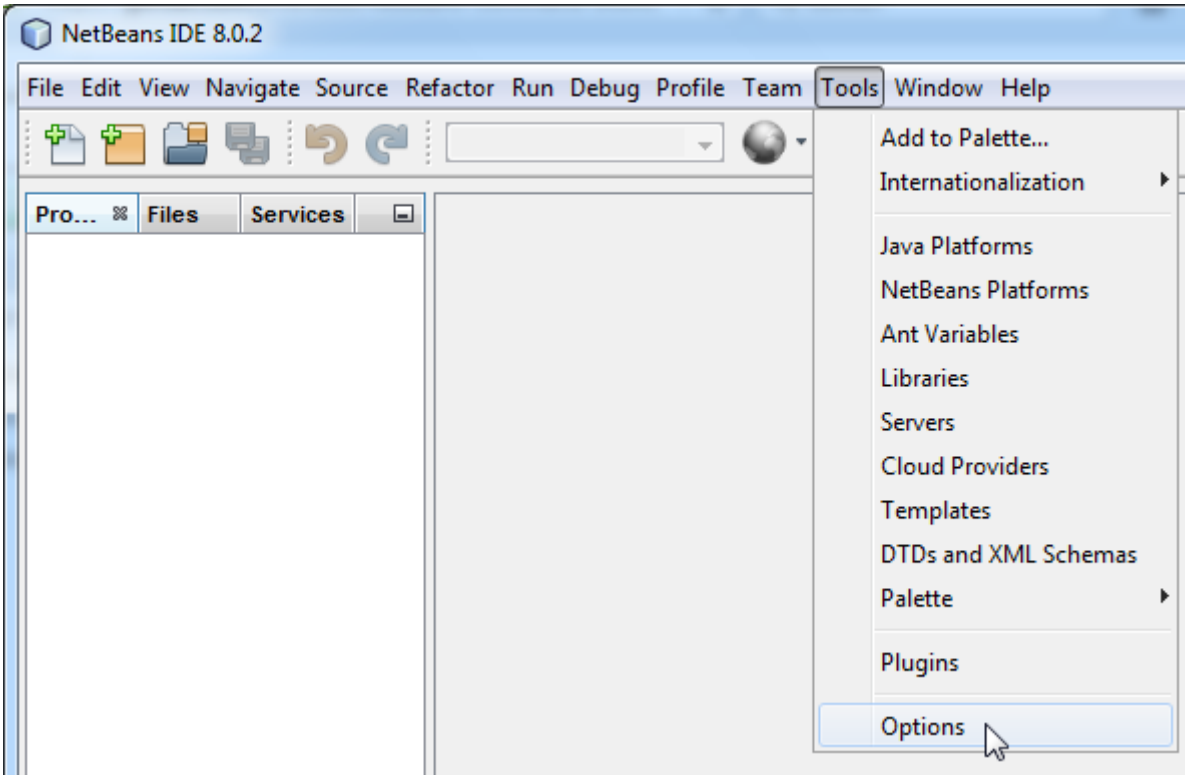
Es wird „Next>“ geklickt.



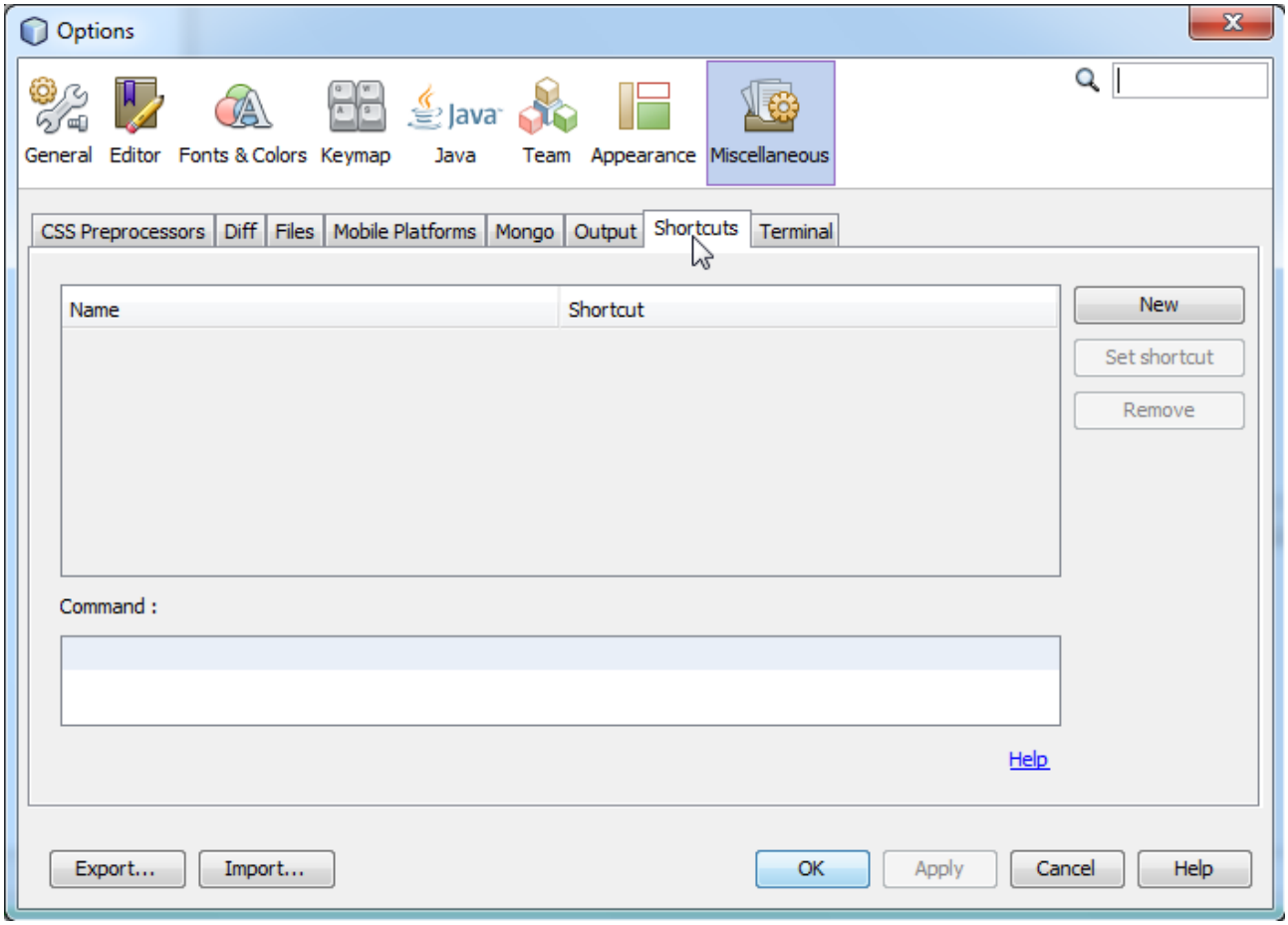
Die weiteren Schritte unterscheiden sich nicht von dem bei der Installation direkt angebotener Plugins, wie sie im Kapitel 15.1 beschrieben wird.

15.5 Nutzung von NetBeans Command Shortcuts

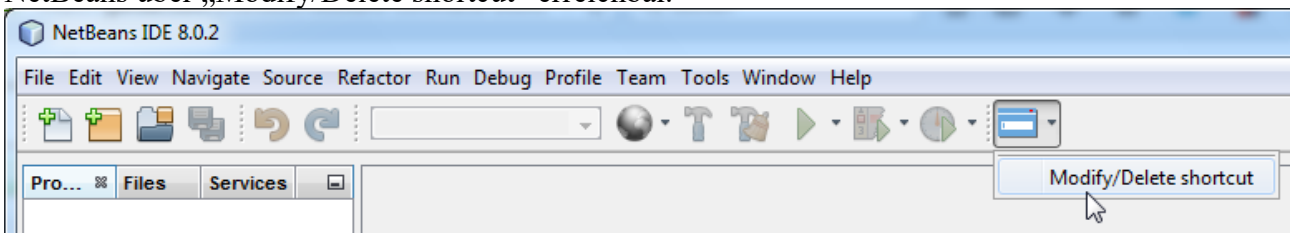
Die Konfigurationsschritte für Plugins sind leider nicht einheitlich, dieses Plugin bietet zwei Versionen an. Der übliche Weg erfolgt über „Tools -> Options“.



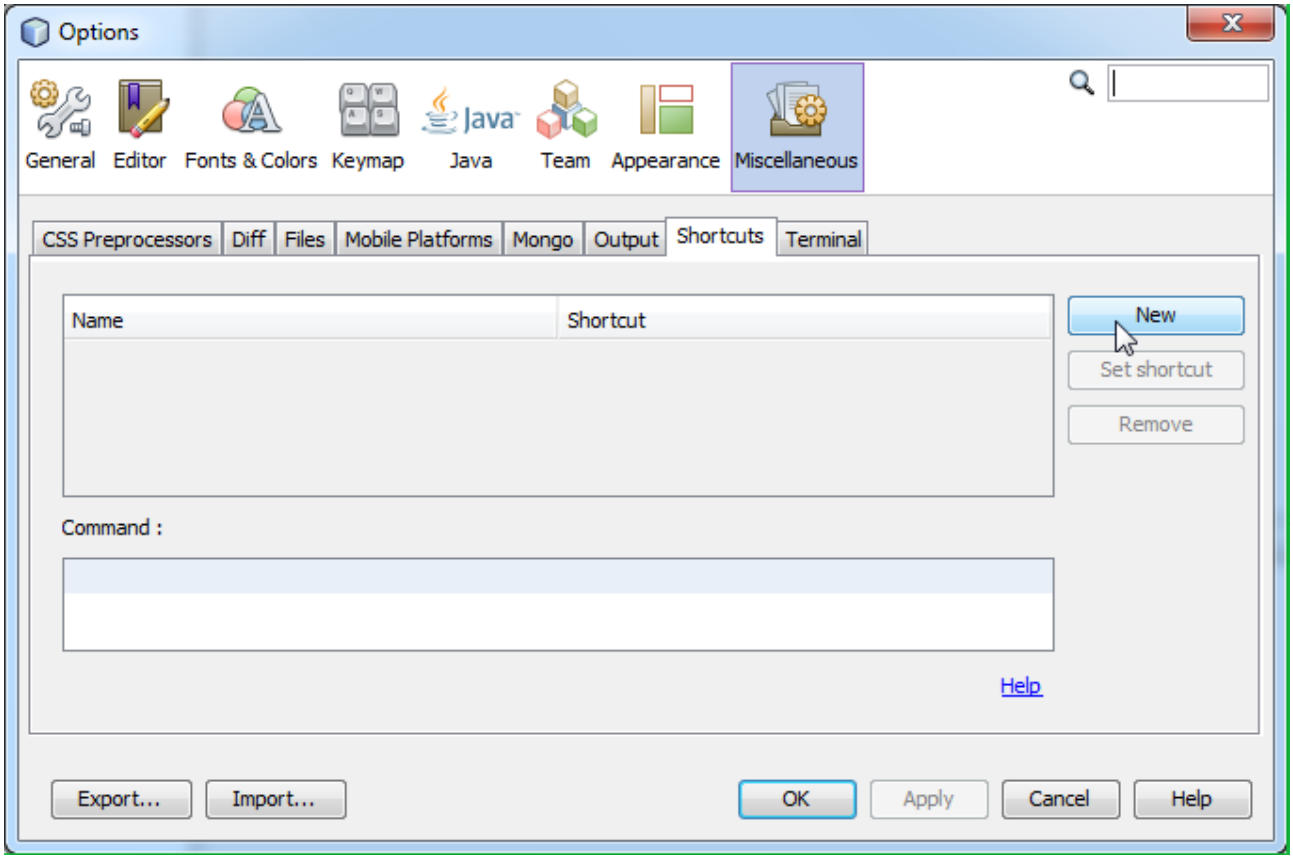
Es wird oben das Icon „Miscellaneous“ und dann der Reiter „Shortcuts“ angeklickt.



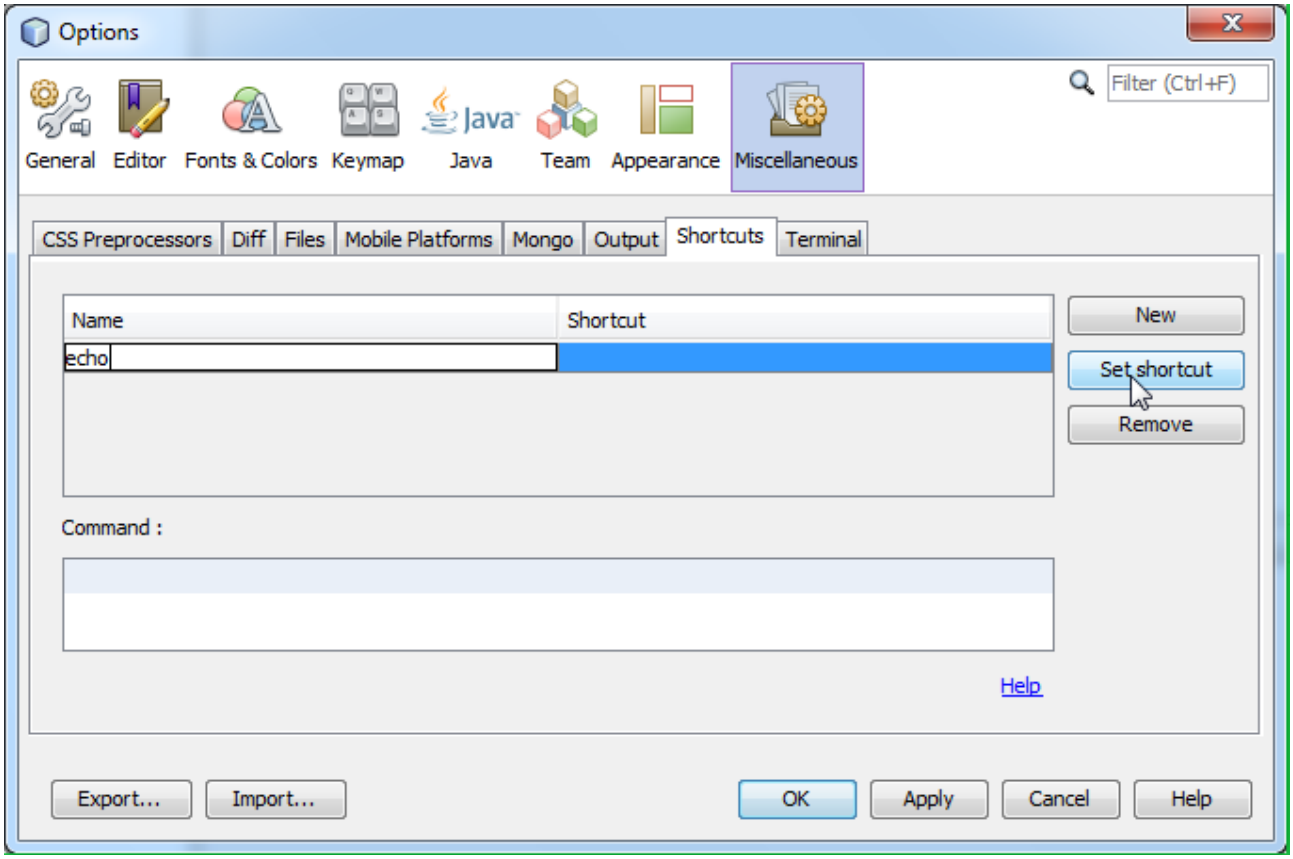
Der gleiche Reiter ist auch direkt über den kleinen Pfeil nach unten direkt neben dem neuen Icon in NetBeans über „Modify/Delete shortcut“ erreichbar.



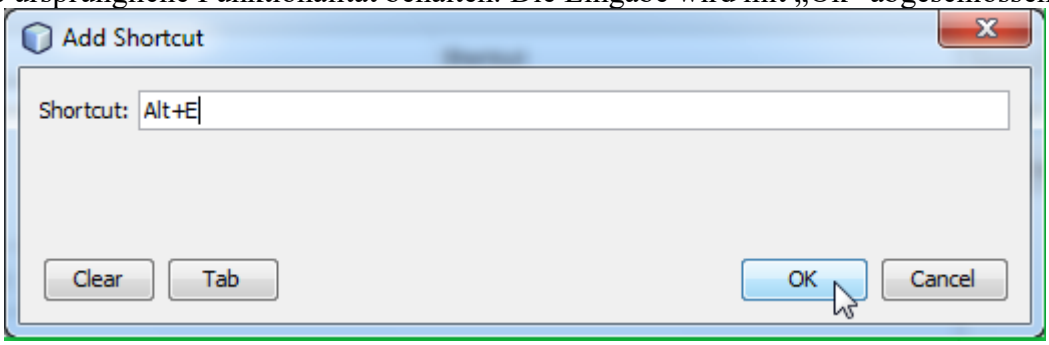
Zur Erstellung eines neuen Shortcuts wird rechts auf „New“ geklickt.



Es wird eine neue Zeile hinzugefügt, in die links der möglichst eindeutige Name des Shortcuts eingetragen wird. Man kann dann rechts auf „Set shortcut“ klicken und ein Tastaturkürzel angeben, dies ist aber nicht zwingend.



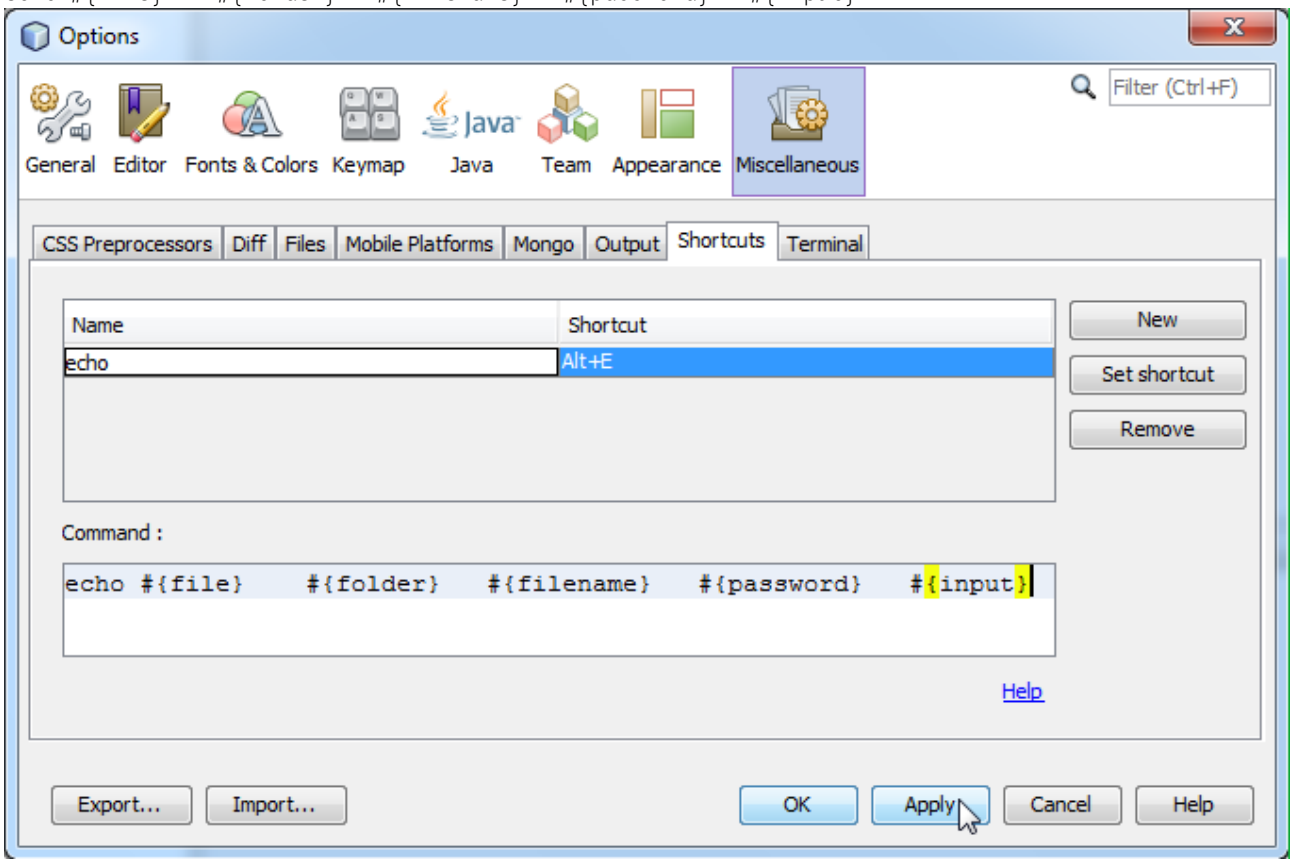
In das Eingabefeld wird dann das Kürzel eingetragen, dabei werden auch Tasten wie „Alt“ und „Strg“ berücksichtigt. Dabei sollten keine Shortcuts genutzt werden, die in NetBeans bereits existieren, da diese ihre ursprüngliche Funktionalität behalten. Die Eingabe wird mit „Ok“ abgeschlossen.



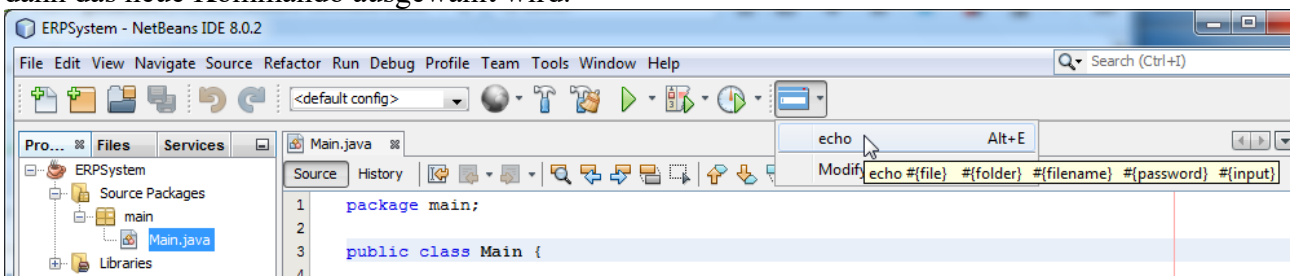
In das untere Command-Feld wird der auszuführende Befehl eingetragen. Im Beispiel ist dies der klassische echo-Befehl aus DOS-Zeiten mit dem einfach ein Text ausgegeben wird. Das Plugin bietet dabei einzelne Zugriffsmöglichkeiten auf die Umgebung, diese sind:
#{file}: gerade selektierte Datei
#{folder}: aktuelles Verzeichnis
#{filename}: Name der akzuell ausgewählten Datei, ohne Endung
#{input}: der Nutzer wird interaktiv aufgefordert einen Text einzugeben
#{password}: der Nutzer wird interaktiv aufgefordert einen Text einzugeben, der maskiert wird.

Will man Programme starten, ist zu beachten, dass vor dem eigentlichen Befehl „`cmd /c`“ steht. Falls der Befehl Leerzeichen enthält, muss der Befehl in Anführungszeichen stehen.
Im Beispiel wird folgender Text als Befehl eingegeben, abschließend die Eingabe mit „Apply“ und „Ok“ abgeschlossen.

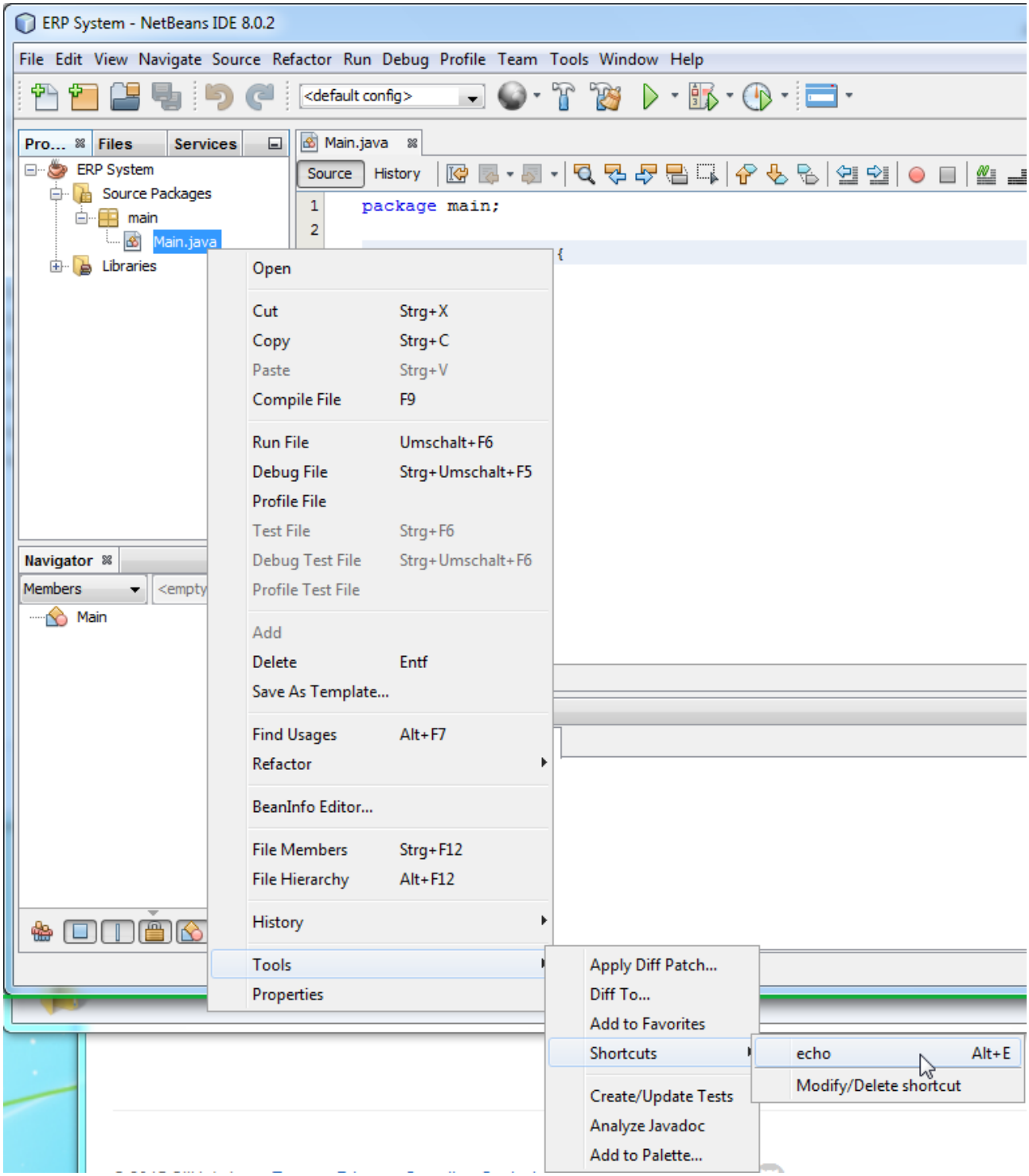
```
echo #{file}    #{folder}    #{filename}    #{password}    #{input}
```



Soll der Befehl ausgeführt werden, wird in einem Projekt zunächst eine Datei ausgewählt. Dann gibt es drei Möglichkeiten. Es kann wieder der kleine Pfeil neben dem neuen Icon geklickt werden, wobei dann das neue Kommando ausgewählt wird.



Alternativ wird nach einem Rechtsklick auf der Datei dann „Tools -> Shortcuts -> echo“ ausgewählt.

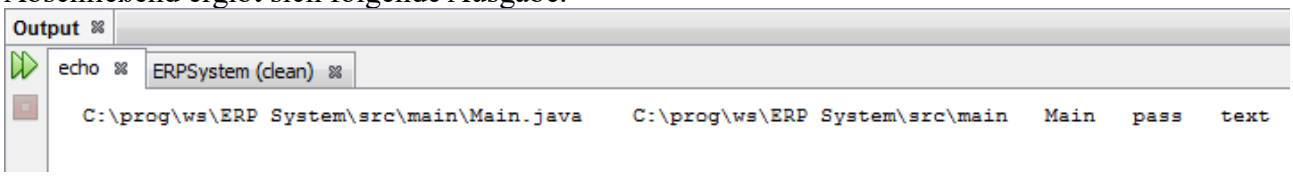


Alternativ kann natürlich auch direkt das angegebene Tastaturkürzel genutzt werden.

Die Ausgabe sind in allen Fällen zunächst zwei Eingabeaufforderungen.



Abschließend ergibt sich folgende Ausgabe.



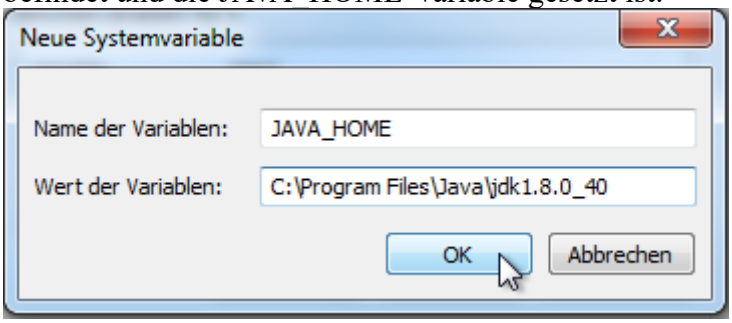
16 MQTT

MQTT ist ein Kommunikationsprotokoll, das z. B. im Bereich des Internet of Things (IoT) zur Kommunikation zwischen Komponenten eingesetzt wird. Der dazu genutzte und auch sonst weit verbreitete Ansatz des Publish-Subcribe-Verfahrens, auch Observer-Obersable-Patterns, kann grundsätzlich auch in anderen Bereichen für den Nachrichtenaustausch genutzt werden. Basis der Übertragung sind einfache Byte-Streams. Für größere Protokolle ist dann eine eigene Umsetzung auf Basis der Byte-Streams zu entwickeln.

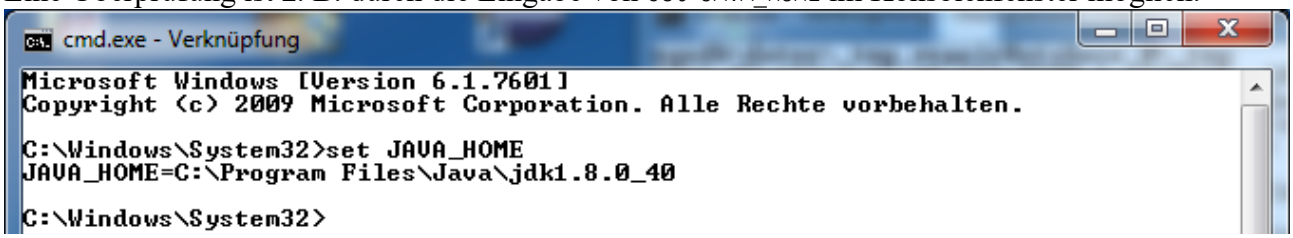
Dieses Kapitel beschreibt die Erstellung eines eigenen MQTT-Brokers, auch als Server oder Kommunikationsplattform bezeichenbar, für eigene Experimente.

16.1 Installation Apache Apollo

Apache Apollo ist ein sogenannter Message Broker, der verschiedene Protokolle, u. a. MQTT unterstützt. Er wird dabei in einem Publish-Subscriber-Ansatz zum Server, bei dem sich Clienten anmelden können, die Nachrichten publizieren wollen und Clienten anmelden, die Nachrichten abonnieren.
Vor einer Installation ist sicherzustellen, dass Java installiert ist, sich das zugehörige bin-Verzeichnis in der Path-Variablen befindet und die JAVA_HOME-Variable gesetzt ist.



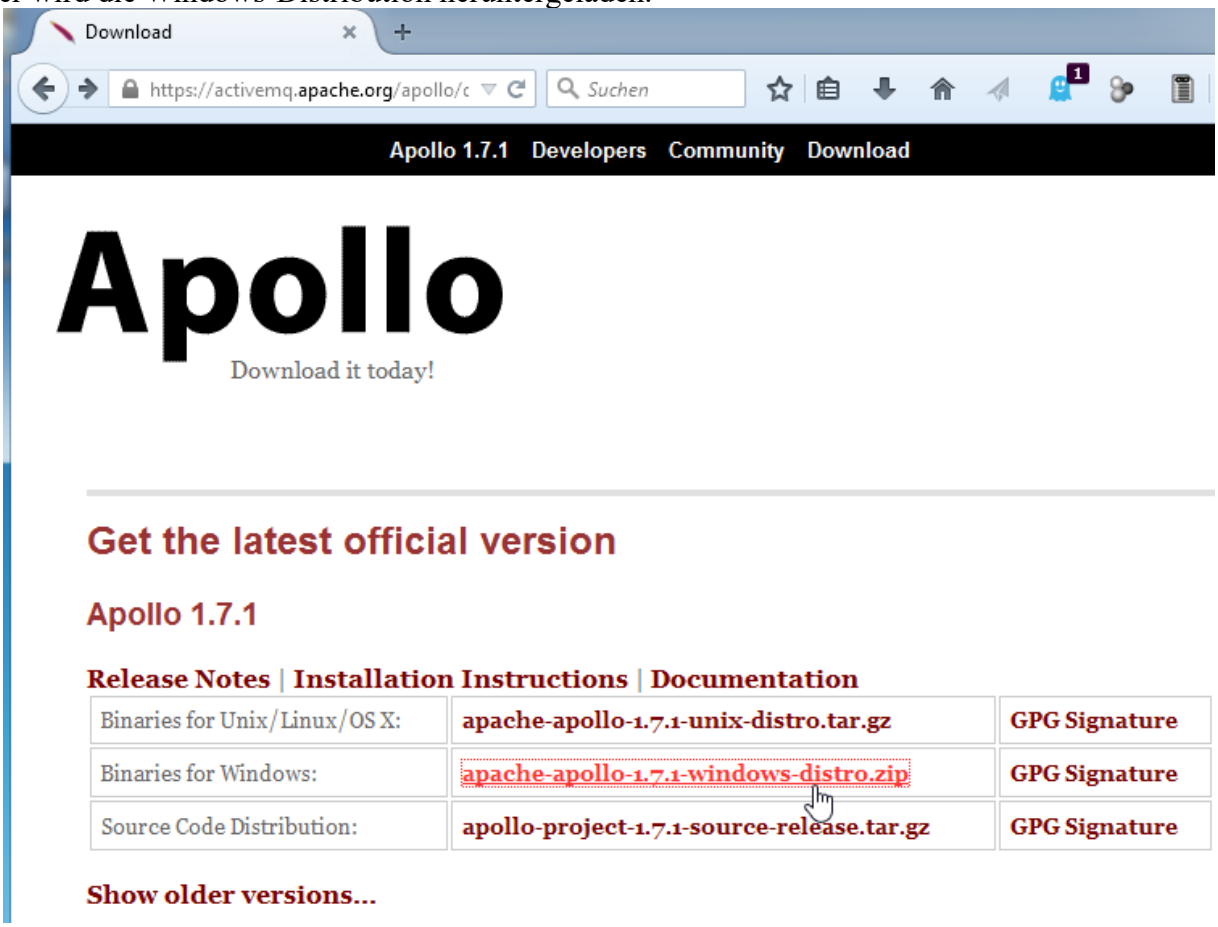
Eine Überprüfung ist z. B. durch die Eingabe von `set JAVA_HOME` im Konsolenfenster möglich.



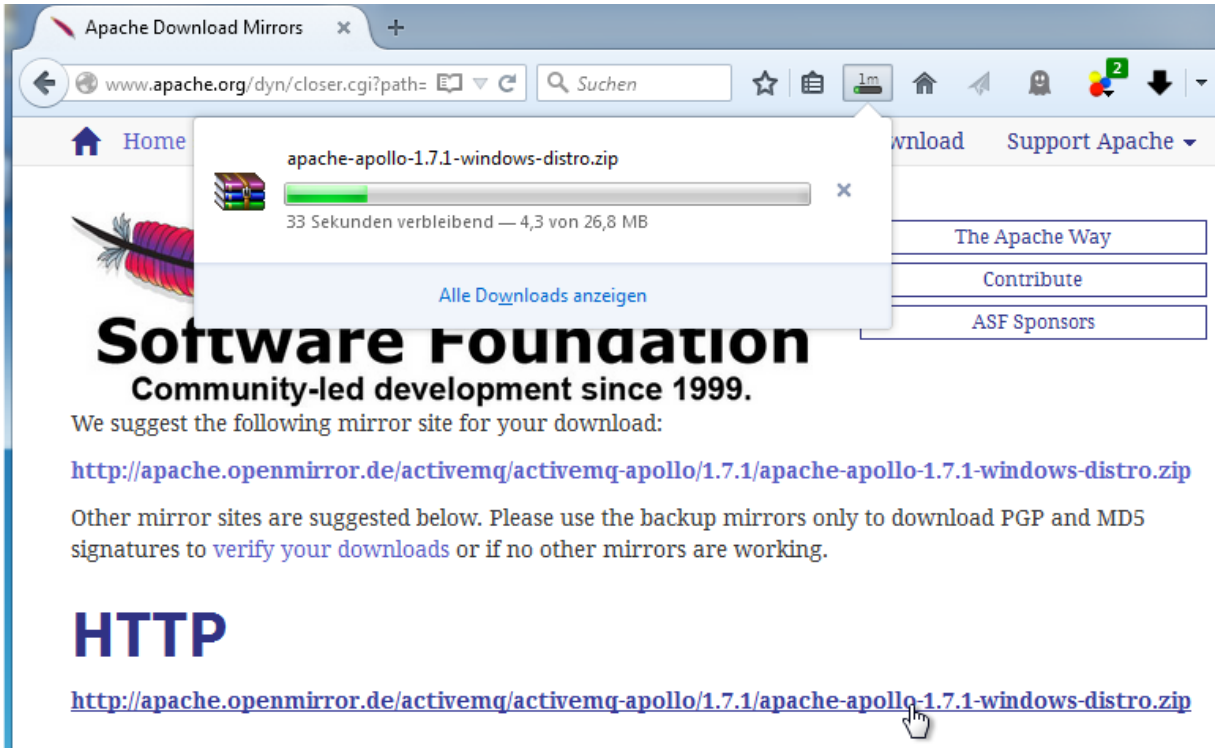
Apollo ist unter der Adresse https://activemq.apache.org/apollo/ erhältlich. Auf der Seite wird oben „Download“ ausgewählt.



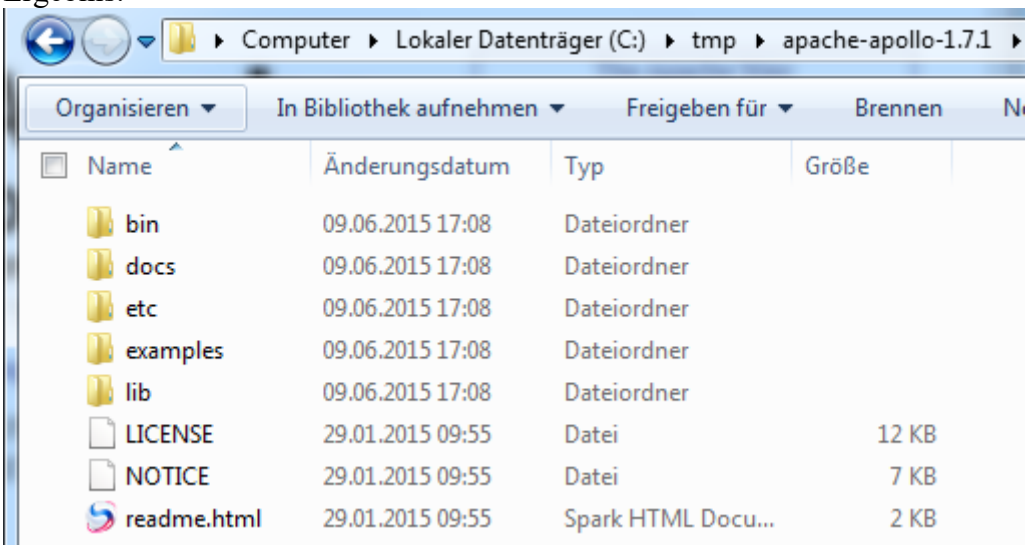
Hier wird die Windows-Distribution heruntergeladen.



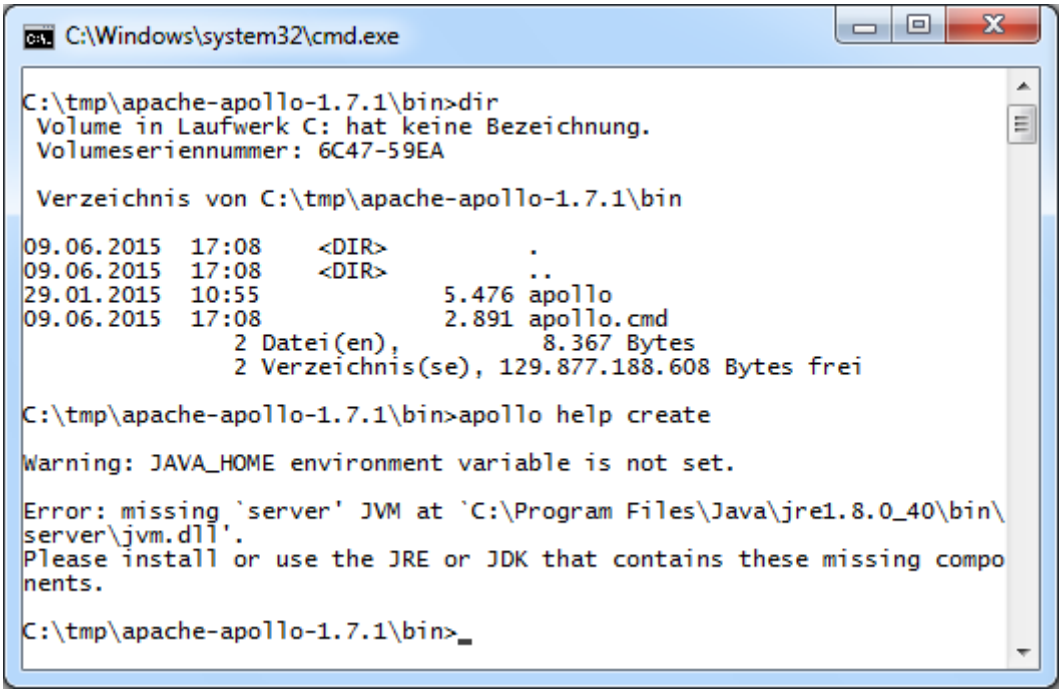
Man kann sich dann eine Web-Seite wie http://apache.openmirror.de/activemq/activemq-apollo/1.7.1/apache-apollo-1.7.1-windows-distro.zip zum Download sich aussuchen.



Die Software wird an einem Ort ausgepackt, an dem man auch Rechte zum Ausführen und Schreiben hat. Hier wird exemplarisch das Verzeichnis C:\tmp gewählt. Man erhält nach dem Auspacken folgendes Ergebnis.



Im bin-Verzeichnis befindet sich die Möglichkeit einen Broker (Server) zu erstellen. Spätestens beim ersten Aufruf würde man auf eine fehlende JAVA_HOME-Variable hingewiesen.



Nun soll eine Instanz des Brokers erstellt werden, was z. B. mit der folgenden Zeile möglich ist, die auch den Entstehungsort des Brokers enthält. In diesem Verzeichnis sind Lese- und Schreibrechte notwendig.

```
apollo create C:\tmp\broker
```

Die Erstellung dauert etwas.

```
C:\tmp\apache-apollo-1.7.1\bin>apollo create C:\tmp\broker
Creating apollo instance at: C:\tmp\broker
Generating ssl keystore...

You can now start the broker by executing:

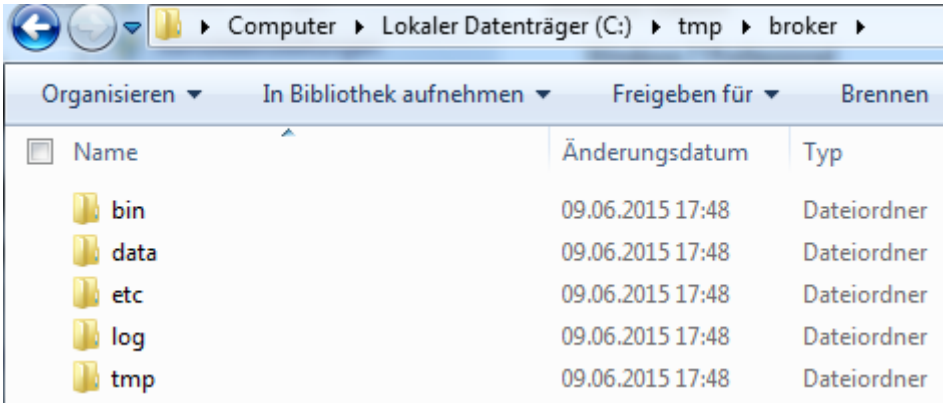
   "C:\tmp\broker\bin\apollo-broker" run

Or you can setup the broker as Windows service and run it in the background:

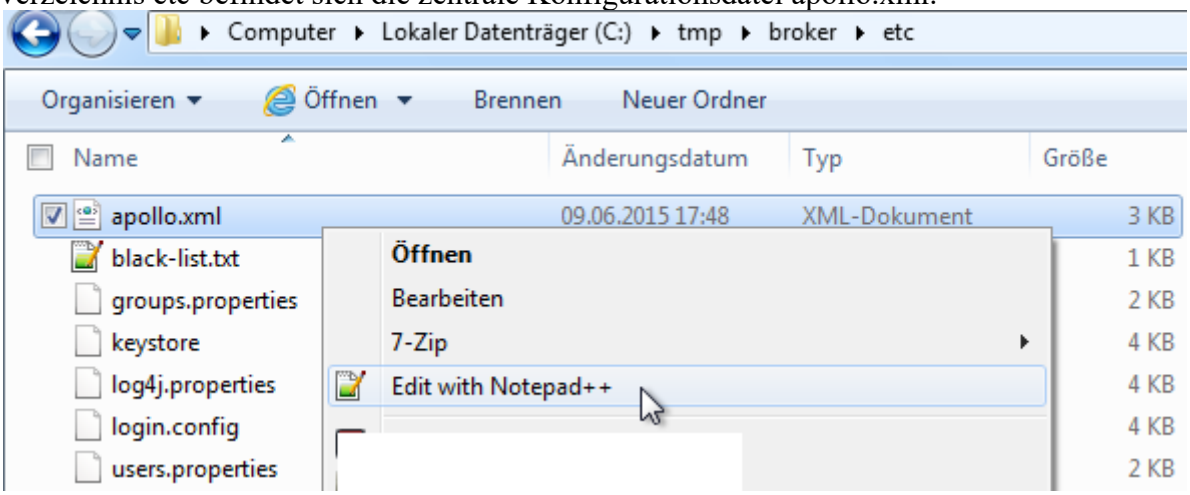
   "C:\tmp\broker\bin\apollo-broker-service" install
   "C:\tmp\broker\bin\apollo-broker-service" start


C:\tmp\apache-apollo-1.7.1\bin>
```

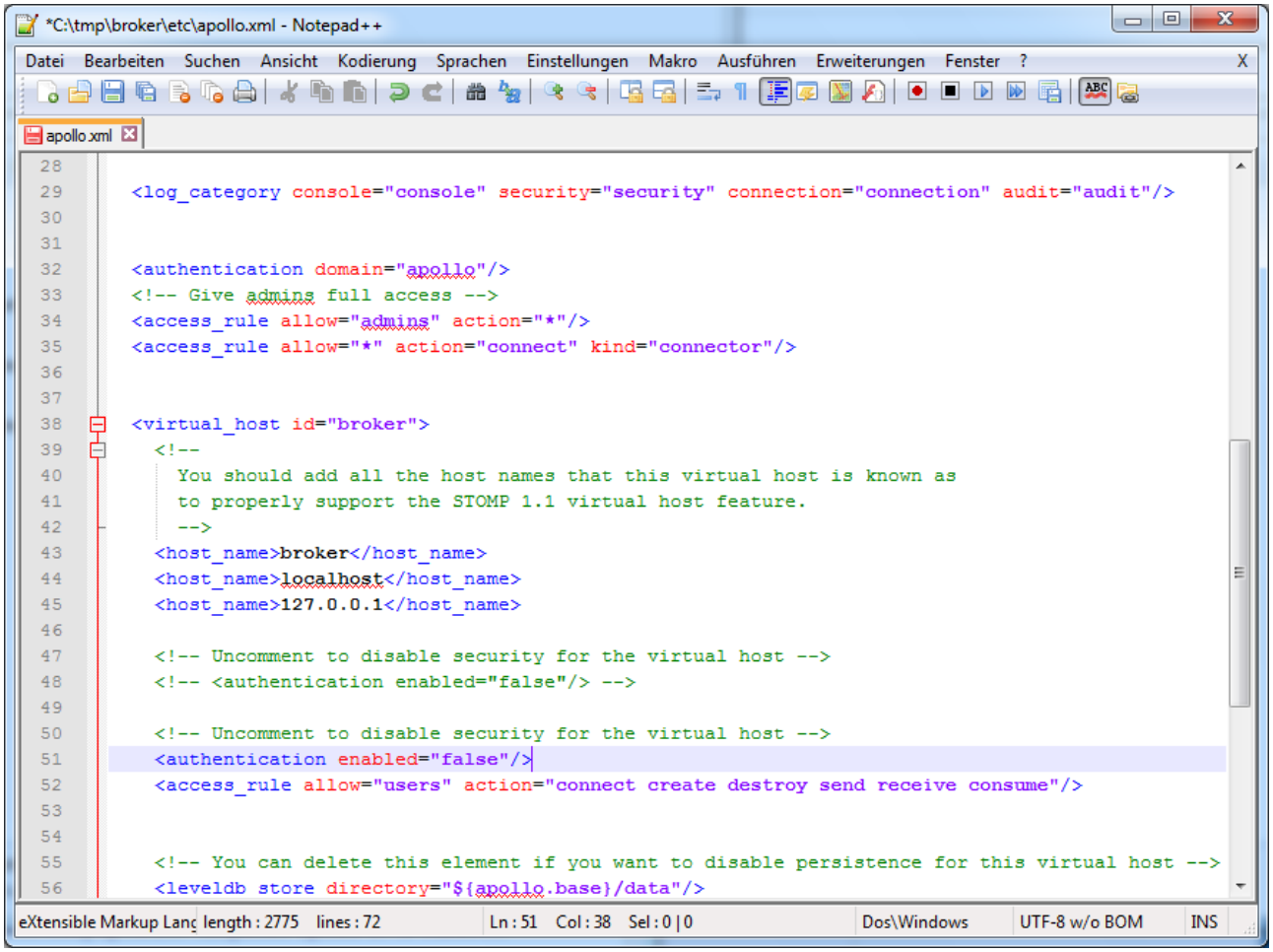
Das Verzeichnis C:\tmp\broker sieht wie folgt aus.



Im Verzeichnis etc befindet sich die zentrale Konfigurationsdatei apollo.xml.

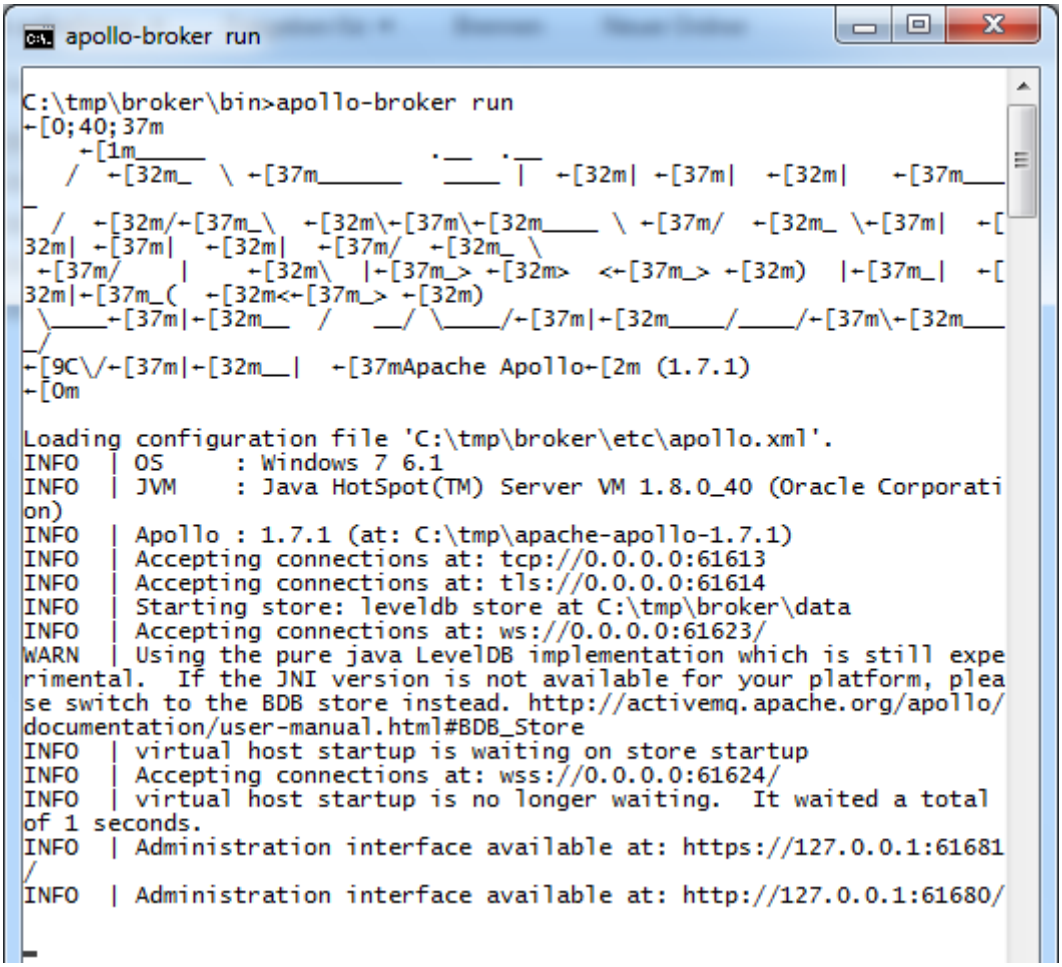


Diese Datei wird zunächst geöffnet und in der Zeile 51 die Kommentierung entfernt. Das Ergebnis zeigt die folgende Abbildung. Durch die Änderung ist es möglich, den Broker ohne Authentifizierung zu nutzen. Generell steht dazu sonst der Nutzer `admin` mit dem Passwort `password` zur Verfügung.

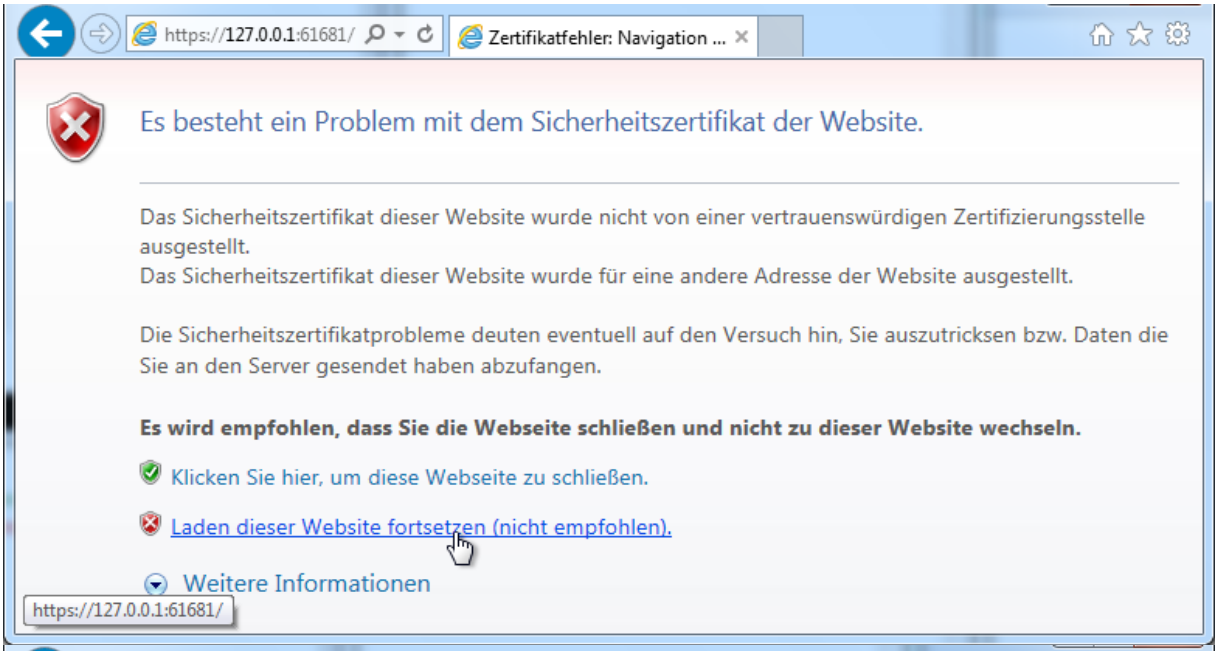


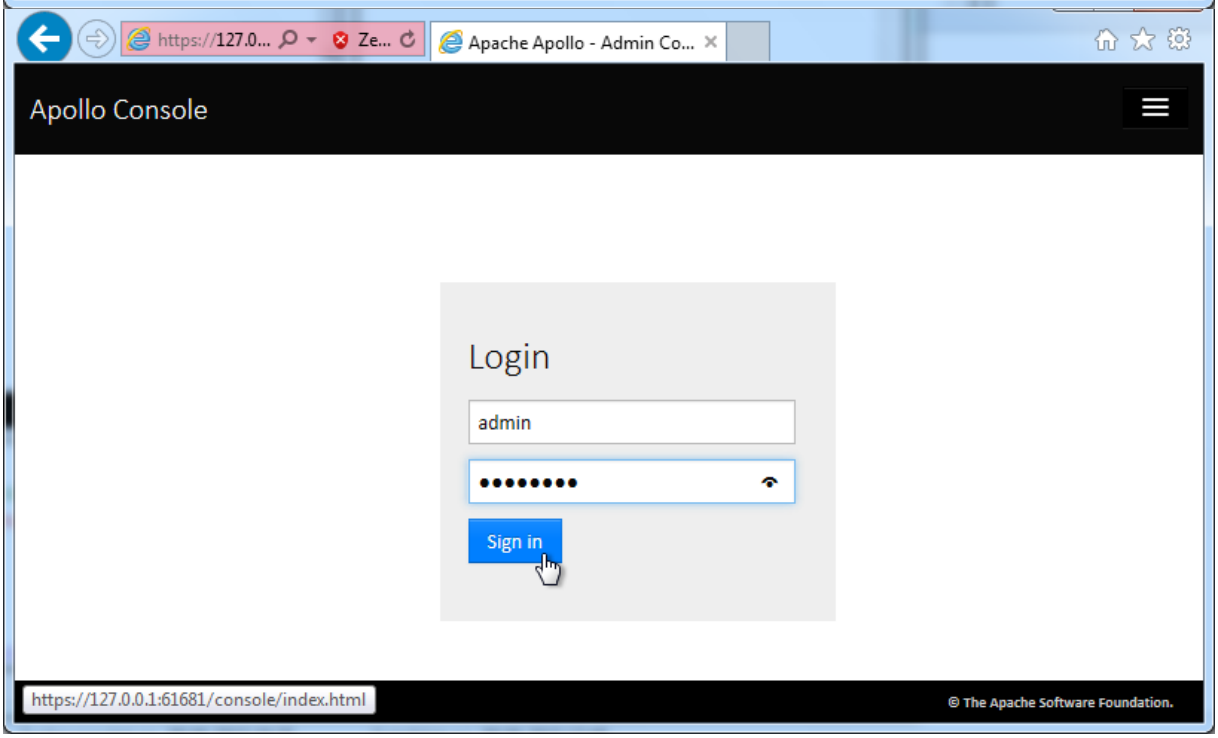
Die Möglichkeit zum Starten des Brokers befindet sich im bin-Verzeichnis. Weiterhin befindet sich hier auch die Konfigurationsdatei apollo-broker-service.xml in der u. a. festgehalten wird, dass diese Instanz von der Basis-Installation abhängig ist. Der Start-Befehl lautet:

```
apollo-broker run
```

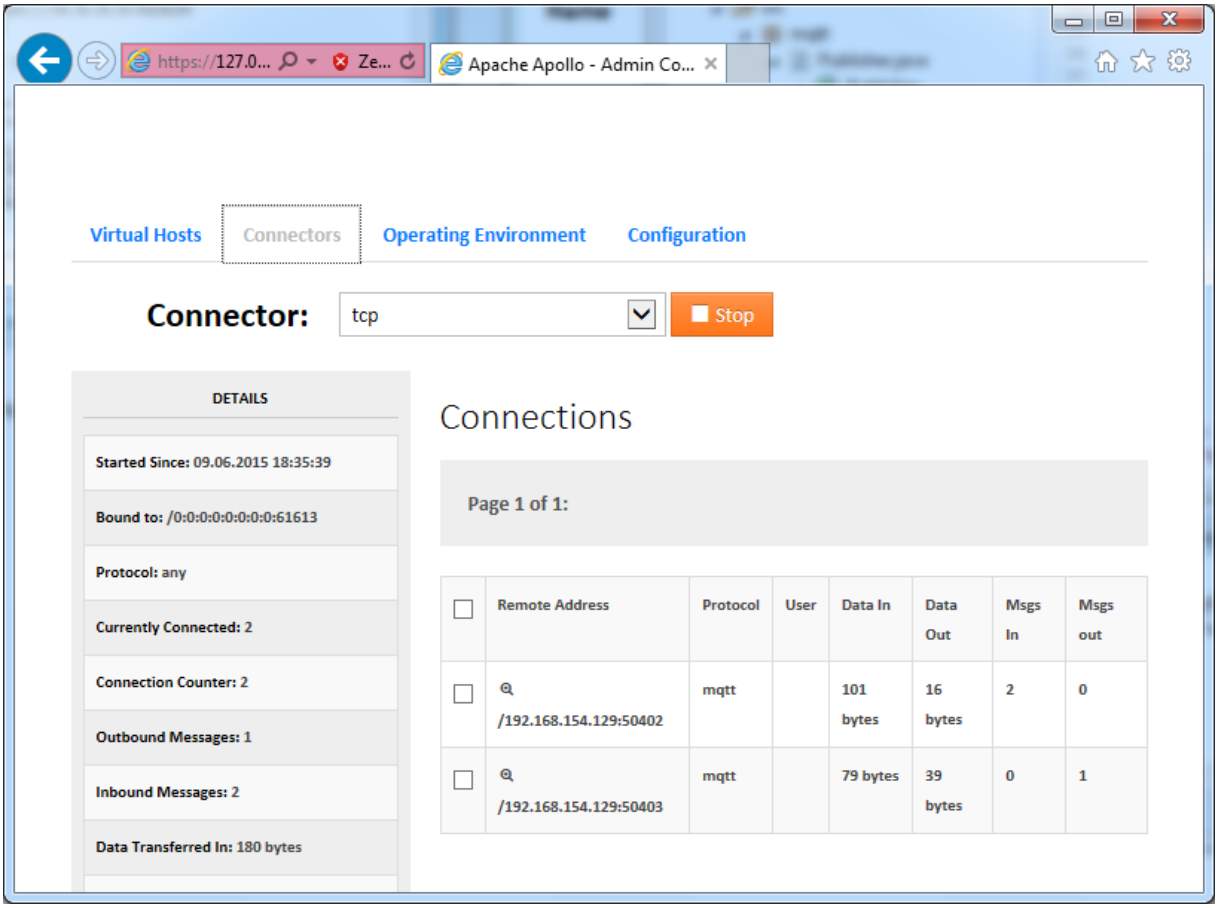


Man beachte, dass dieses Fenster geöffnet bleiben muss, solange der Broker laufen soll. Die Variante, den Broker als System-Dienst laufen zu lassen, wird hier nicht betrachtet. Das vorherige Fenster gibt u. a. die relevanten Informationen zum Verbindungsaufbau bekannt. Man kann sich z. B. davon überzeugen, dass der Server läuft, indem im Browser die Adresse https://127.0.0.1:61681/ eingegeben wird. Man landet auf dem Startbildschirm, bei dem man sich als admin (user:admin, password:password), nachdem man der Verbindung vertraut, anmelden kann.

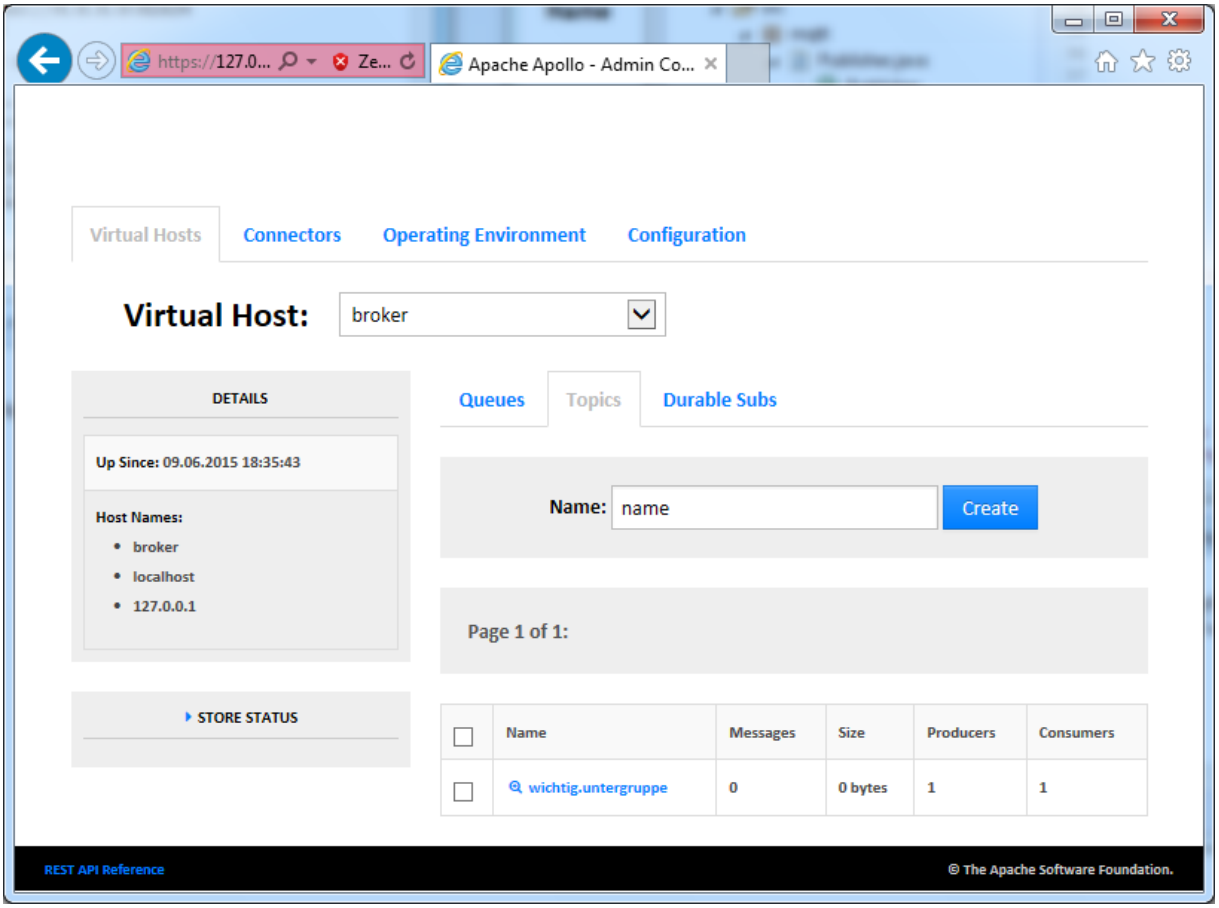




Die Begrüßungsseite enthält einige Basisinformationen. Im laufenden Betrieb ist es sinnvoll, oben den Reiter Connectors auszuwählen, da hier aktive Verbindungen beobachtet werden können.



Weiterhin kann der Reiter Topics im unteren Bereich über die bearbeiteten Themen informieren. Man kann so erkennen, ob erfolgreich eine Verbindung aufgebaut wurde und Themen zur Kommunikation zur Verfügung stehen.

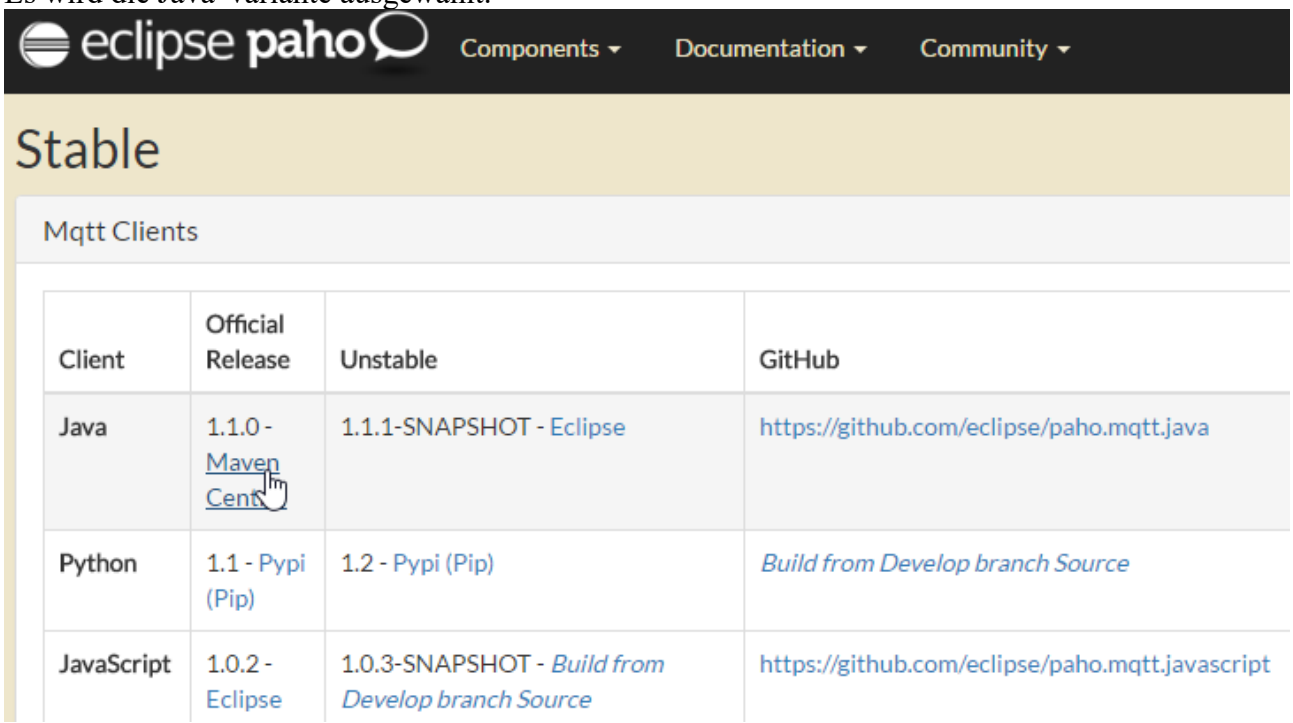


16.2 Client-Entwicklung mit Eclipse paho

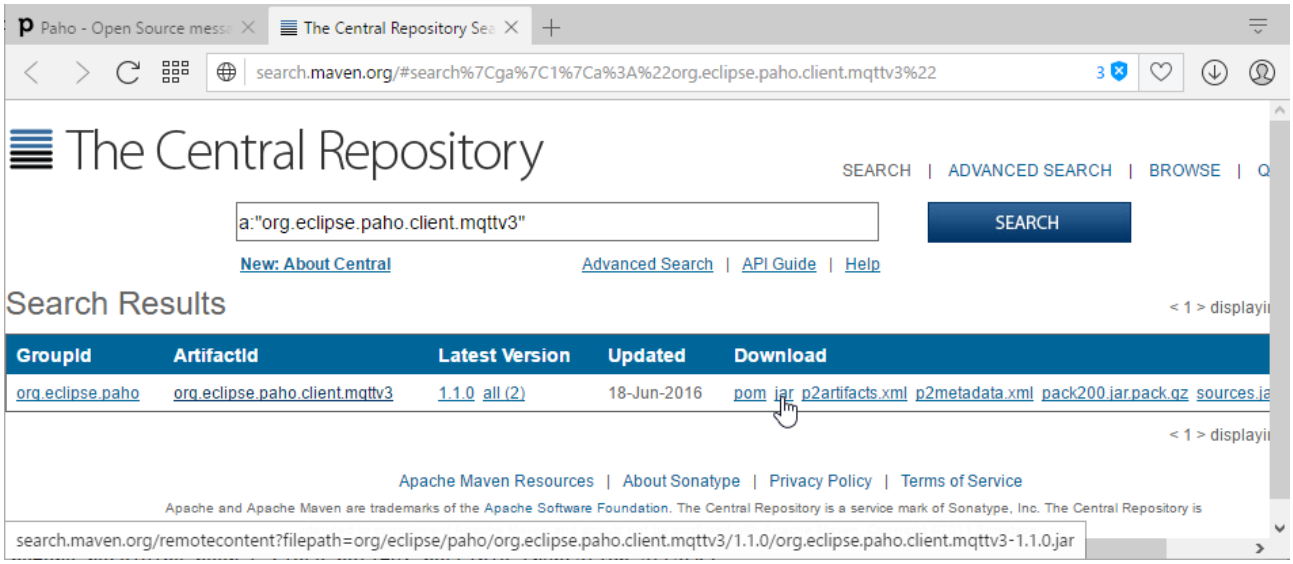
Auch zur Entwicklung von Clients steht eine große Menge von Möglichkeiten für sehr unterschiedliche Programmiersprachen und Spezialhardware, wie Arduino, zur Verfügung. Hier wird die etablierte Bibliothek des Eclipse paho-Projekts https://eclipse.org/paho/ genutzt. Es wird auf „Download Now“ geklickt.



Es wird die Java-Variante ausgewählt.

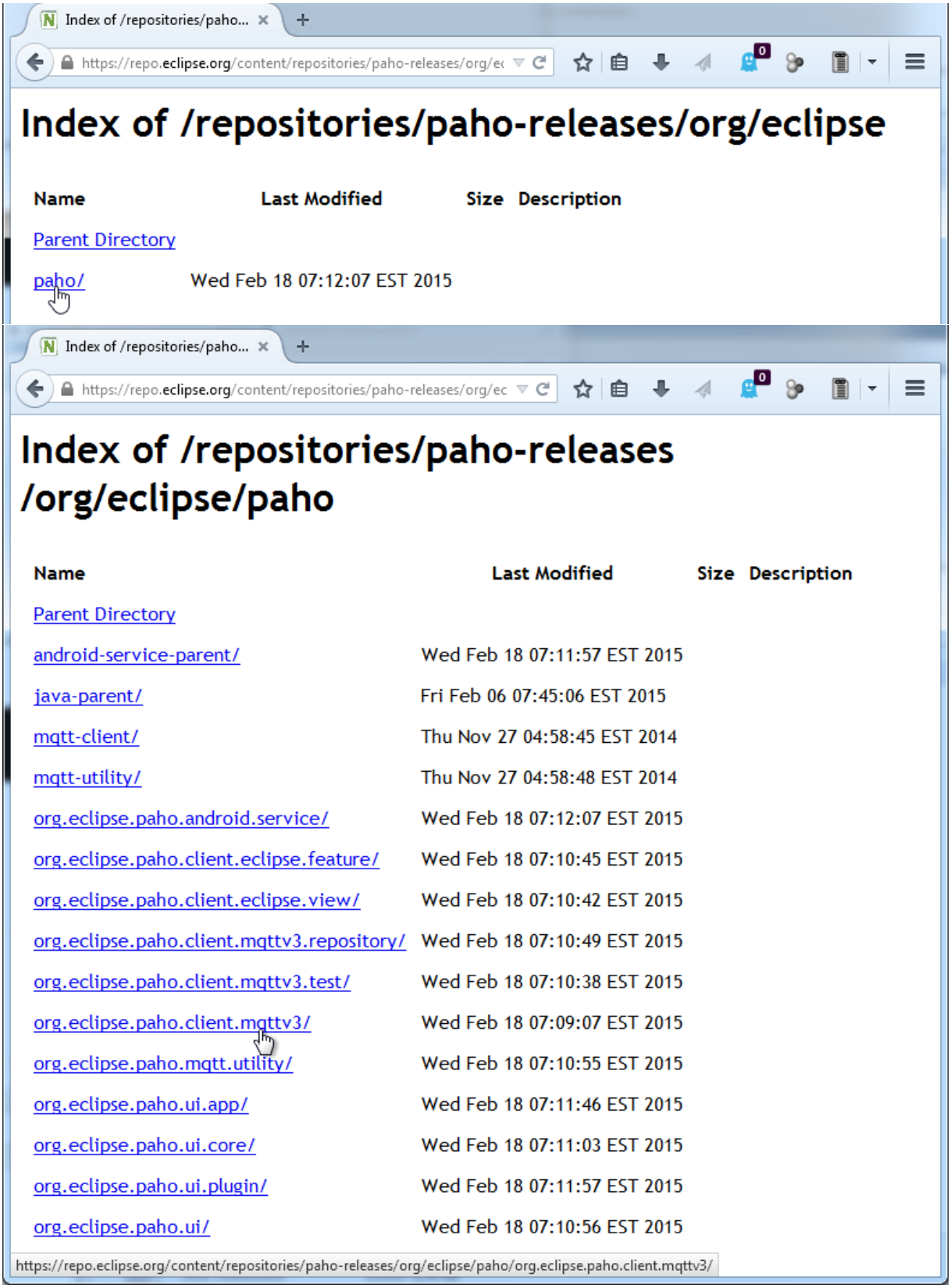


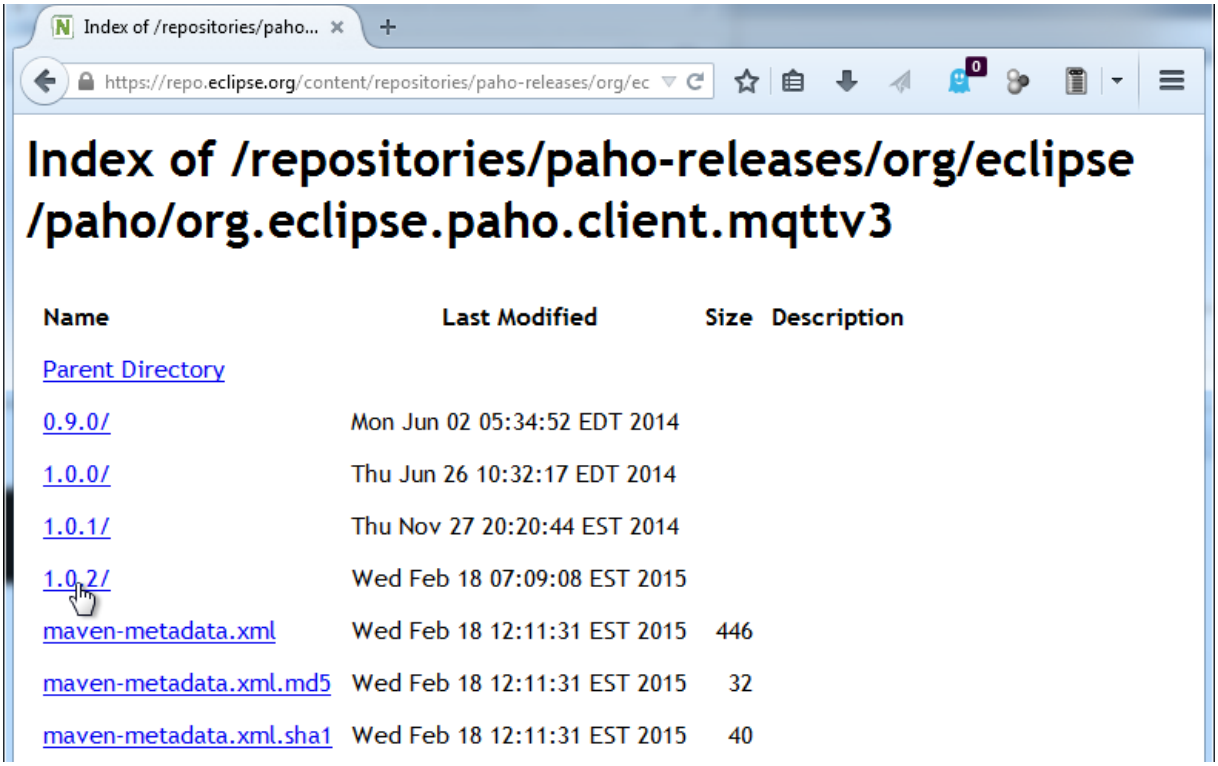
Es wird die Jar-Datei heruntergeladen.



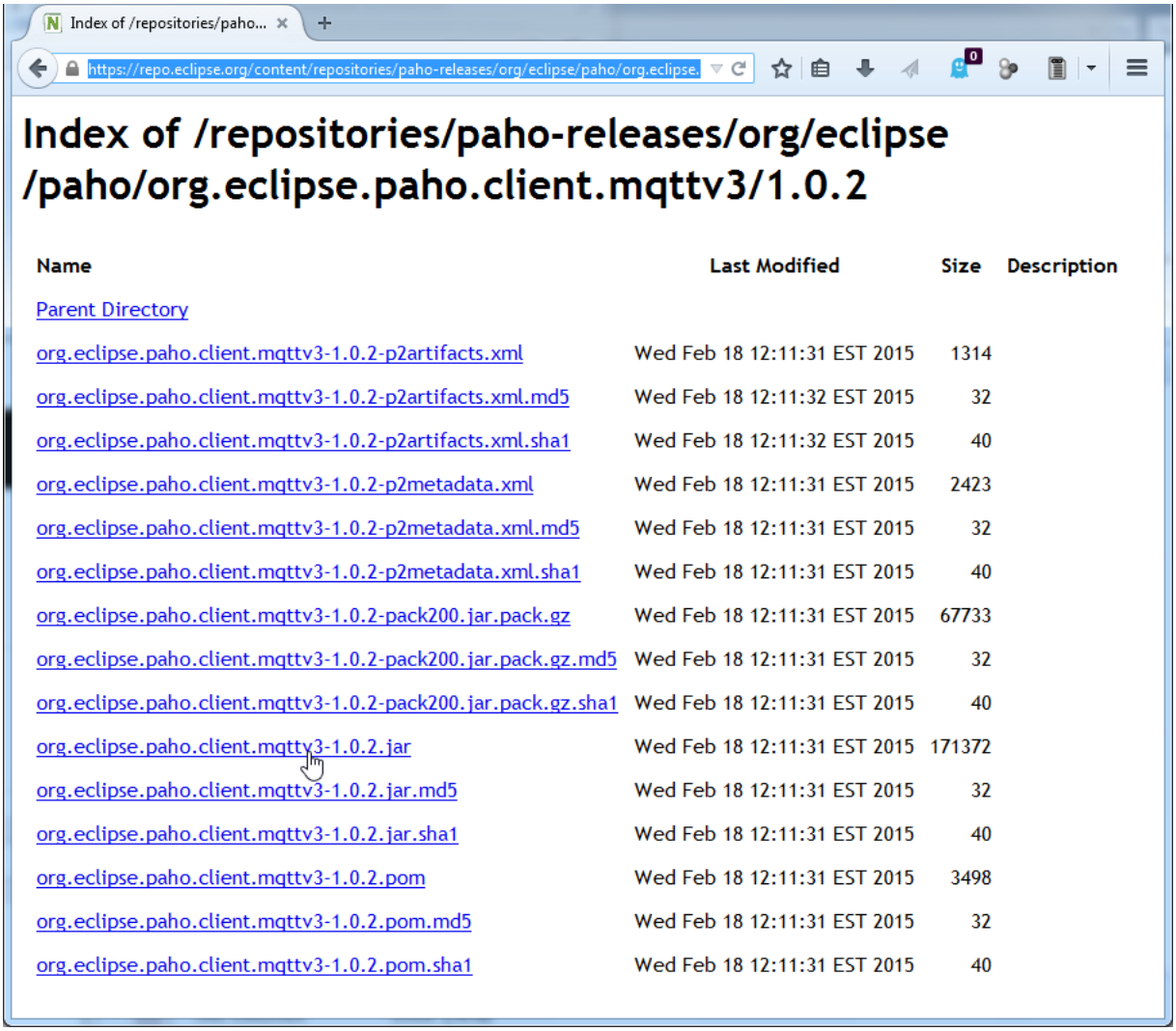
Alternativ kann zum Download die Seite https://repo.eclipse.org/content/repositories/paho-releases/ genutzt werden. Hier hangelt man sich durch Klicks zunächst zum Ordner org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.0.2 oder einer aktuelleren Vwersion herunter.




Index of /repositories/paho...
https://repo.eclipse.org/content/repositories/paho-releases/org/ec
Index of /repositories/paho-releases/org/eclipse
Name Last Modified Size Description
Parent Directory
paho/ Wed Feb 18 07:12:07 EST 2015
Index of /repositories/paho...
https://repo.eclipse.org/content/repositories/paho-releases/org/ec
Index of /repositories/paho-releases /org/eclipse/paho
Name Last Modified Size Description
Parent Directory
android-service-parent/ Wed Feb 18 07:11:57 EST 2015
java-parent/ Fri Feb 06 07:45:06 EST 2015
mqtt-client/ Thu Nov 27 04:58:45 EST 2014
mqtt-utility/ Thu Nov 27 04:58:48 EST 2014
org.eclipse.paho.android.service/ Wed Feb 18 07:12:07 EST 2015
org.eclipse.paho.client.eclipse.feature/ Wed Feb 18 07:10:45 EST 2015
org.eclipse.paho.client.eclipse.view/ Wed Feb 18 07:10:42 EST 2015
org.eclipse.paho.client.mqttv3.repository/ Wed Feb 18 07:10:49 EST 2015
org.eclipse.paho.client.mqttv3.test/ Wed Feb 18 07:10:38 EST 2015
org.eclipse.paho.client.mqttv3/ Wed Feb 18 07:09:07 EST 2015
org.eclipse.paho.mqtt.utility/ Wed Feb 18 07:10:55 EST 2015
org.eclipse.paho.ui.app/ Wed Feb 18 07:11:46 EST 2015
org.eclipse.paho.ui.core/ Wed Feb 18 07:11:03 EST 2015
org.eclipse.paho.ui.plugin/ Wed Feb 18 07:11:57 EST 2015
org.eclipse.paho.ui/ Wed Feb 18 07:10:56 EST 2015
https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.client.mqttv3/



In dem Ordner https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.0.2/ kann dann die benötigte Jar-Datei org.eclipse.paho.client.mqttv3-1.0.2.jar heruntergeladen werden.



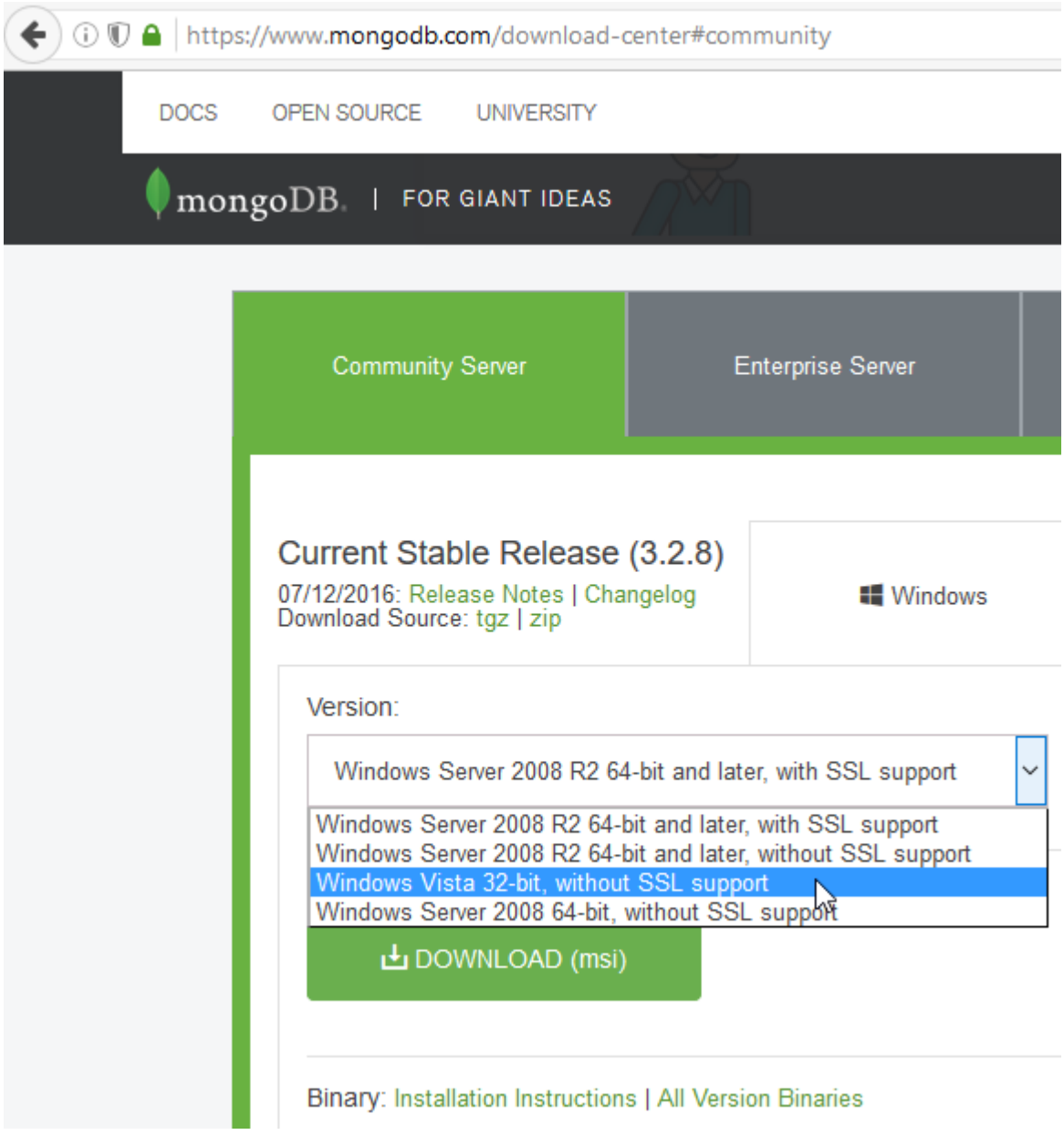
Die Jar-Datei wird dann normal in die Projekte, z. B. als Kopie in einem lib-Verzeichnis, eingebunden.

17 MongoDB

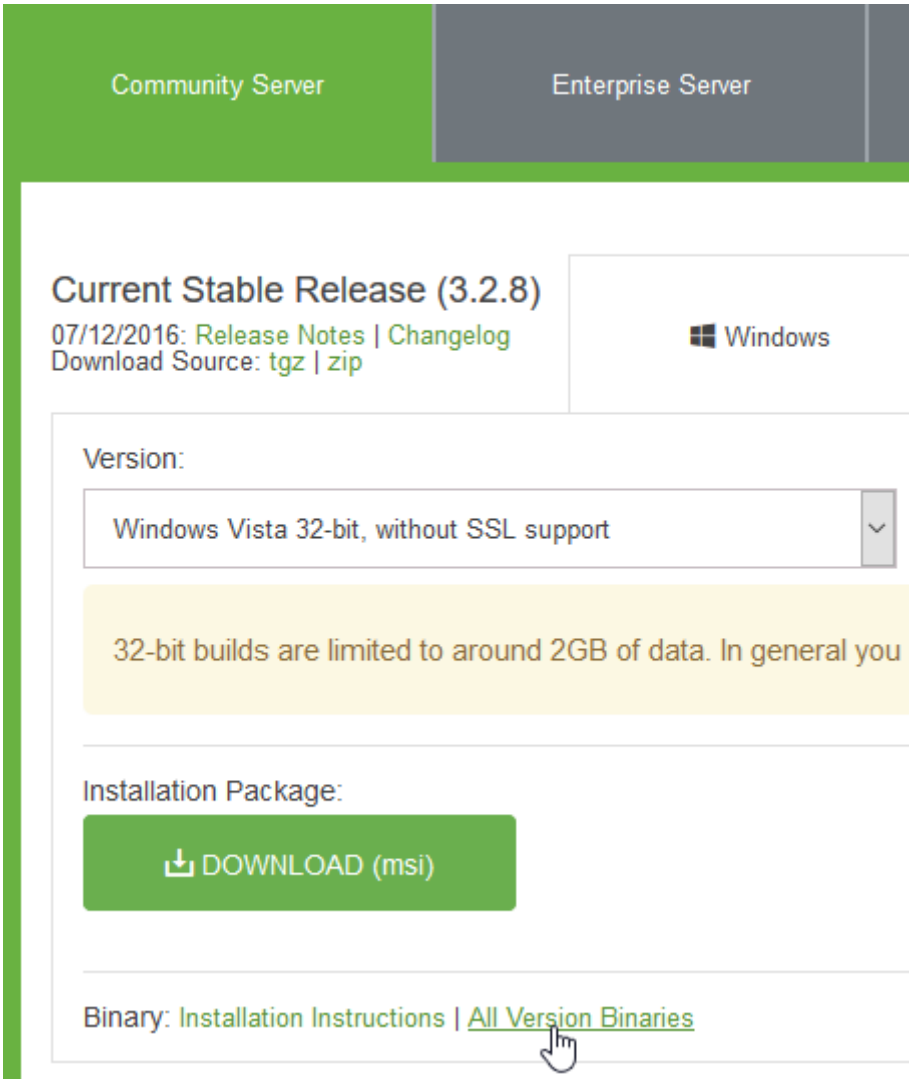
Neben den klassischen relationalen Datenbanken gibt es eine heterogene Gruppe von sogenannten NoSQL-Datenbanken. Dabei steht „NoSQL“ für „not only SQL“, was bereits andeutet, dass Konzepte von SQL keine oder eine geringe Rolle spielen. Eine wesentliche Motivation von NoSQL-Datenbanken ist, dass zusammengehörige Daten in relationalen Datenbanken oft über viele Tabellen verteilt sind. Der zunächst willkürliche Begriff „zusammengehörig“ ist dabei etwas durch „Daten, die sehr häufig immer zusammen genutzt werden“ präzisierbar. Durch die Idee, zusammengehörige Daten auch zusammen kompakt zu speichern, kommt es zu einem Ansatz von NoSQL-Datenbanken, deren prominenteste Vertretung die MongoDB (http://www.mongodb.org/) ist, deren Name sich aus dem englischen Begriff „humongous“ für „gigantisch“ ableitet. Generell stellen damit NoSQL-Datenbanken eine Alternative zu relationalen Datenbanken dar, die für bestimmte Aufgaben klare Vorteile bieten, aber generell nicht das Ziel der Verdrängung relationaler Datenbanken haben. In großen Projekten mit fast disjunkten Datenbereichen muss individuell über die Nutzung der passenden Datenbankfunktionalität nachgedacht werden, da ein Zusammenspiel verschiedener Datenbanken mit unterschiedlicher Technik möglich ist und sinnvoll sein kann.

17.1 Installation

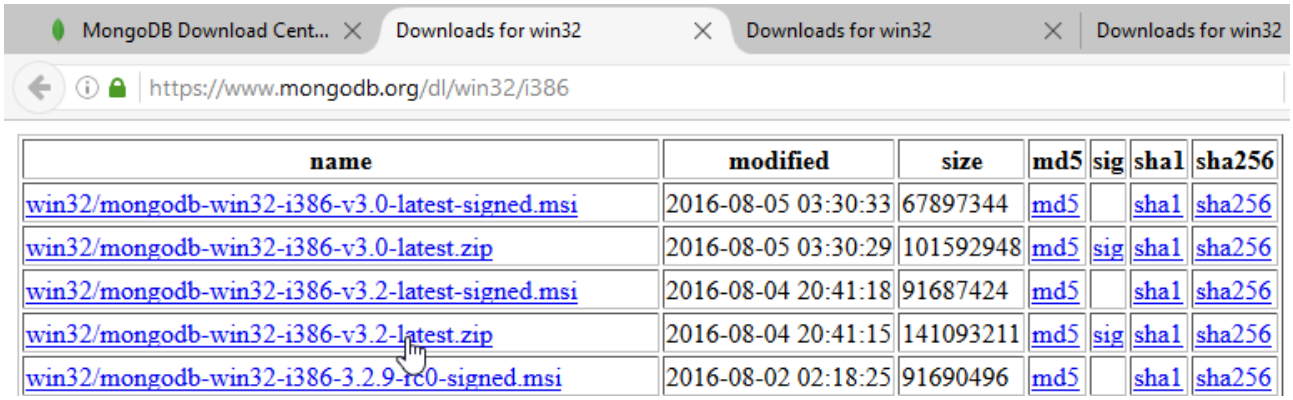
Hier werden nur die ersten Schritte mit MongoDB unter Windows beschrieben. Der Download kann über die Seite https://www.mongodb.com/download-center#community starten, auf der zunächst „Versions“ eine passende Version, hier „Windows Vista“ ausgewählt wird. Bisher war es so, dass es sich um eine Aussage „ab Windows Vista lauffähig“ handelt.



Es wird auf „All Version Binaries“ geklickt. Es öffnet sich danach https://www.mongodb.org/dl/win32/i386.

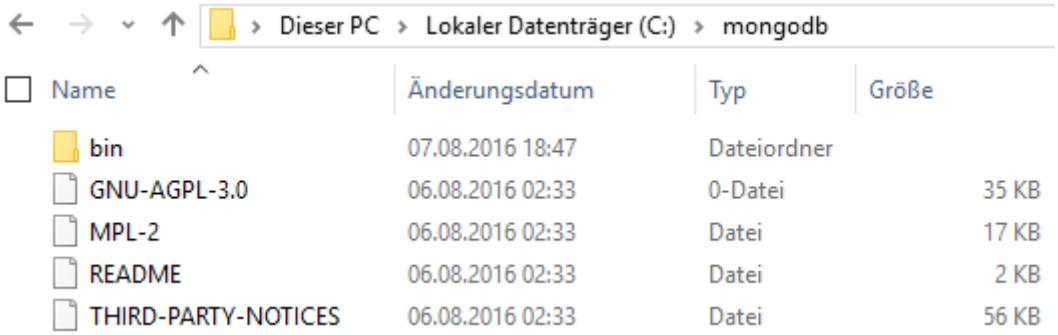


Es wird hier ein „Zip-File für Windows“ genutzt, dass zur 32-bit- oder 64-bit-Variante des Betriebssystems passen sollte. Die benötigte Datei ist hier für 64-bit http://downloads.mongodb.org/win32/mongodb-win32-x86_64-2008plus-ssl-latest.zip. Man beachte, dass mit der 32-bit-Variante nur deutlich kleinere Datenmengen verwaltet werden können. Der MSI-Installer ist alternativ auch nutzbar. Bei der 32-bit-Variante müssen beim Start der Datenbank einige Details beachtet werden, die aus der zugehörigen Dokumentation entnehmbar sind.



name	modified	size	md5	sig	sha1	sha256
win32/mongodb-win32-i386-v3.0-latest-signed.msi	2016-08-05 03:30:33	67897344	md5		sha1	sha256
win32/mongodb-win32-i386-v3.0-latest.zip	2016-08-05 03:30:29	101592948	md5	sig	sha1	sha256
win32/mongodb-win32-i386-v3.2-latest-signed.msi	2016-08-04 20:41:18	91687424	md5		sha1	sha256
win32/mongodb-win32-i386-v3.2-latest.zip	2016-08-04 20:41:15	141093211	md5	sig	sha1	sha256
win32/mongodb-win32-i386-3.2.9-rc0-signed.msi	2016-08-02 02:18:25	91690496	md5		sha1	sha256

Die heruntergeladene Datei kann in einem beliebigen Verzeichnis, hier nach einer Namenskürzung C:\mongodb, entpackt werden.



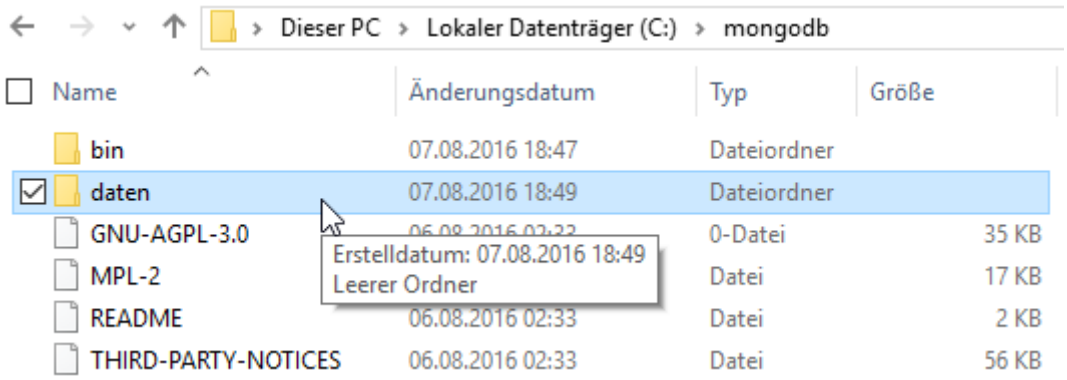
Die Datenbank-Software ist damit nutzbar.

17.2 Start der Datenbank

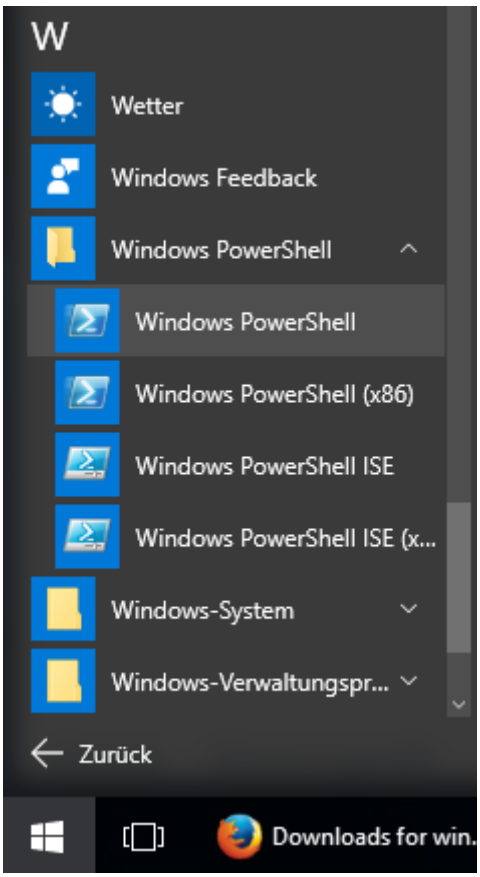
Es gibt eine größere Anzahl von Werkzeugen, die die Arbeit mit MongoDB erleichtern, die allerdings hier nicht vorgestellt werden. Der Fokus liegt auf dem Kommandozeilen-Werkzeug, mit dem erste Schritte sehr einfach ausführbar sind.

Als Vorbereitung muss ein neues Verzeichnis angelegt werden, in dem MongoDB die Daten speichert. Dieses Verzeichnis sollte am Anfang leer sein und für keine anderen Daten genutzt werden.

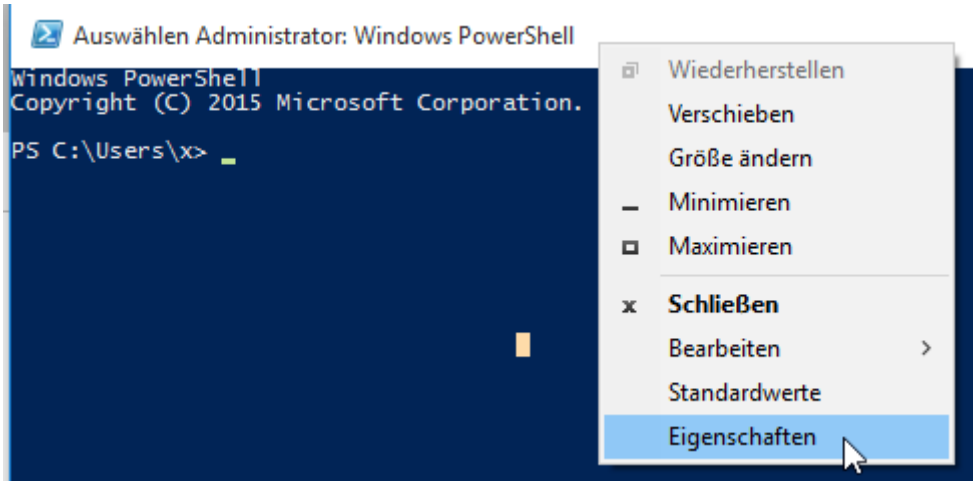
Im Beispiel wird zum Experimentieren ein Ordner C:\mongodb\daten angelegt, wobei es generell sehr unüblich ist, diesen Ordner im Installationsverzeichnis zu platzieren.



Zum Start wird ein Kommandozeilenfenster benötigt, was man z. B. unter dem Start-Knopf bekommt, wenn „Alle Apps“ und dann unter „W“ das Programm „Windows Powershell“ angeklickt wird. Das klassische cmd-Line-Werkzeug (cmd.exe) von Windows kann auch genutzt werden.



Mit einem Rechtklick auf dem oberen Balken können unter „Eigenschaften“ u. a. der Font und die Schriftgröße geändert werden, was hier aber nicht weiter betrachtet wird.



Es wird in das Installationsverzeichnis von MongoDB, genauer das zugehörige bin-Verzeichnis gesteuert.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. Alle Rechte vorbehalten.

PS C:\Users\x> cd C:\mongodb\bin
PS C:\mongodb\bin> dir


    Verzeichnis: C:\mongodb\bin


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        07.08.2016     19:40        4380672 bsondump.exe
-a----        07.08.2016     19:40        7740928 mongo.exe
------        04.08.2016     16:52       67342336 mongo.pdb
-a----        07.08.2016     19:40       14069760 mongod.exe
------        04.08.2016     16:57      153473024 mongod.pdb
-a----        07.08.2016     19:40        9457152 mongodump.exe
-a----        07.08.2016     19:40        5873152 mongoexport.exe
-a----        07.08.2016     19:40        5734400 mongofiles.exe
-a----        07.08.2016     19:40        5974528 mongoimport.exe
-a----        07.08.2016     19:40        5490688 mongooplog.exe
-a----        07.08.2016     19:40       11912192 mongoperf.exe
------        04.08.2016     16:57      143093760 mongoperf.pdb
-a----        07.08.2016     19:40       13778944 mongorestore.exe
-a----        07.08.2016     19:40        6254592 mongos.exe
------        04.08.2016     16:56       85504000 mongos.pdb
-a----        07.08.2016     19:40        5724160 mongostat.exe
-a----        07.08.2016     19:40        5561344 mongotop.exe

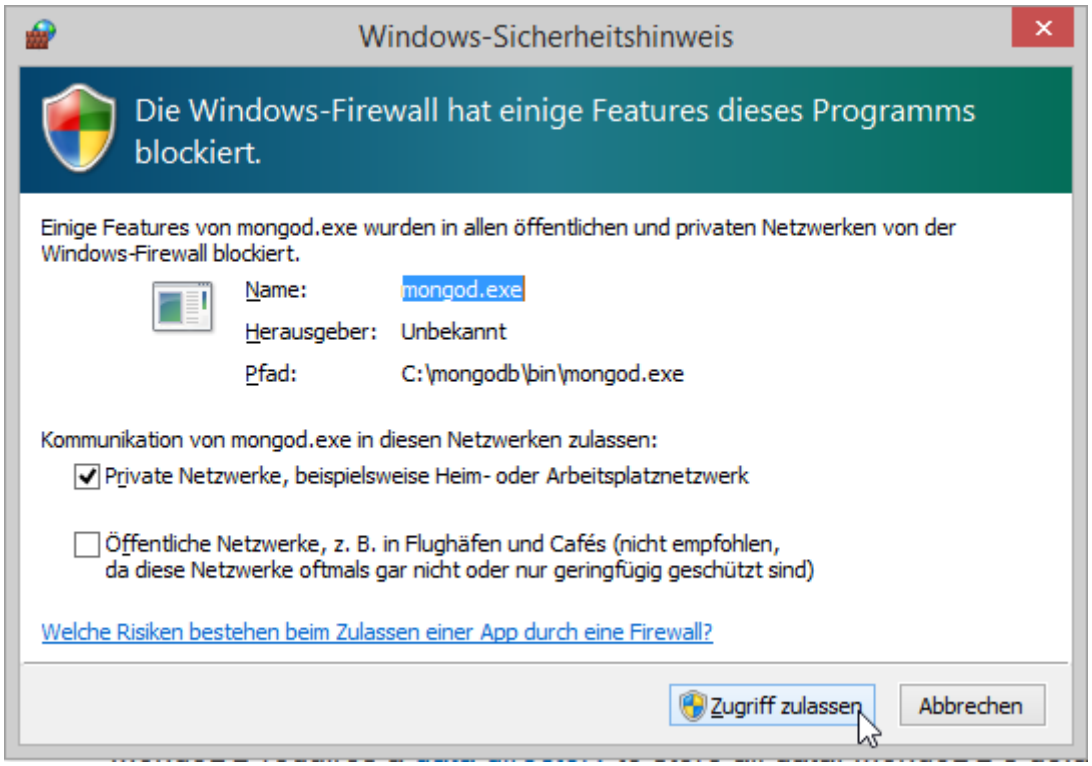

PS C:\mongodb\bin>
```

Zum Start der Datenbank wird das Programm mongod aufgerufen und als einziger Parameter mit „--dbpath" das Verzeichnis mit den Daten angegeben. Die Anführungsstriche können weggelassen werden, wenn keine Leerzeichen im Pfad enthalten sind. Der letze Parameter „storageEngine" muss nur beim ersten Aufruf, also beim Erzeugen der Datenbank angegeben werden.

```
.\mongod --dbpath "c:\mongodb\daten" --storageEngine=nmapv1
```

```
PS C:\mongodb\bin> .\mongod --dbpath "c:\mongodb\daten" --storageEngine=mmapv1
2016-08-07T19:45:40.958+0200 I CONTROL  [main]
2016-08-07T19:45:40.959+0200 W CONTROL  [main] 32-bit servers don't have journaling enabled by default. Please use --journal if you want durability.
2016-08-07T19:45:40.959+0200 I CONTROL  [main]
2016-08-07T19:45:40.972+0200 I CONTROL  [initandlisten] MongoDB starting : pid=7640 port=27017 dbpath=c:\mongodb\daten 32-bit host=DESKTOP-MA7HQL0
2016-08-07T19:45:40.973+0200 I CONTROL  [initandlisten] targetMinOS: Windows Vista/Windows Server 2008
2016-08-07T19:45:40.973+0200 I CONTROL  [initandlisten] db version v3.2.9-rc0-5-ga5191eb
2016-08-07T19:45:40.973+0200 I CONTROL  [initandlisten] git version: a5191ebf7d94e4b04bfccffbac56bd95bc9ec98b
2016-08-07T19:45:40.974+0200 I CONTROL  [initandlisten] allocator: tcmalloc
2016-08-07T19:45:40.975+0200 I CONTROL  [initandlisten] modules: none
2016-08-07T19:45:40.976+0200 I CONTROL  [initandlisten] build environment:
2016-08-07T19:45:40.977+0200 I CONTROL  [initandlisten]     distarch: i386
2016-08-07T19:45:40.977+0200 I CONTROL  [initandlisten]     target_arch: i386
2016-08-07T19:45:40.979+0200 I CONTROL  [initandlisten] options: { storage: { dbPath: "c:\mongodb\daten", engine: "mmapv1" } }
2016-08-07T19:45:40.992+0200 I CONTROL  [initandlisten]
2016-08-07T19:45:40.992+0200 I CONTROL  [initandlisten] ** WARNING: This 32-bit MongoDB binary is deprecated
2016-08-07T19:45:40.993+0200 I CONTROL  [initandlisten] **      NOTE: This is a 32-bit MongoDB binary running on a 64-bit operating
2016-08-07T19:45:40.994+0200 I CONTROL  [initandlisten] **            system. Switch to a 64-bit build of MongoDB to
2016-08-07T19:45:40.995+0200 I CONTROL  [initandlisten] **            support larger databases.
2016-08-07T19:45:40.996+0200 I CONTROL  [initandlisten]
2016-08-07T19:45:41.000+0200 I CONTROL  [initandlisten]
2016-08-07T19:45:41.002+0200 I CONTROL  [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2016-08-07T19:45:41.006+0200 I CONTROL  [initandlisten] **       32 bit builds are limited to less than 2GB of data (or less with --journal).
2016-08-07T19:45:41.006+0200 I CONTROL  [initandlisten] **       Note that journaling defaults to off for 32 bit and is currently off.
2016-08-07T19:45:41.010+0200 I CONTROL  [initandlisten] **       See http://dochub.mongodb.org/core/32bit
2016-08-07T19:45:41.014+0200 I CONTROL  [initandlisten]
2016-08-07T19:45:41.016+0200 I INDEX    [initandlisten] allocating new ns file c:\mongodb\daten\local.ns, filling with zeroes...
2016-08-07T19:45:41.135+0200 I STORAGE  [FileAllocator] allocating new datafile c:\mongodb\daten\local.0, filling with zeroes...
2016-08-07T19:45:41.135+0200 I STORAGE  [FileAllocator] creating directory c:\mongodb\daten\_tmp
2016-08-07T19:45:41.137+0200 I STORAGE  [FileAllocator] done allocating datafile c:\mongodb\daten\local.0, size: 64MB,  took 0.001 secs
2016-08-07T19:45:41.142+0200 I NETWORK  [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2016-08-07T19:45:41.142+0200 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'c:/mongodb/daten/diagnostic.data'
2016-08-07T19:45:41.144+0200 I NETWORK  [initandlisten] waiting for connections on port 27017
```

Abhängig von der installierten Sicherheits-Software müssen ggfls. einige Rechte eingeräumt werden.



Nun ist die Datenbank gestartet. Das Fenster muss, solange die Datenbank genutzt wird, geöffnet bleiben. Bei einem professionelleren Einsatz kann der Datenbankstart als Windows-Service

konfiguriert werden, so dass auch ein unmittelbarer Start in Verbindung mit dem Hochfahren des Rechners möglich ist. Weiterhin ist es sinnvoll, das bin-Verzeichnis der MongoDB in die Pfad-Variable des Rechners einzutragen.

Wird vergessen das Datenbankfenster vor dem Herunterfahren des Rechners zu schließen, kann die Datenbank in einen inkonsistenten Zustand geraten, was beim nächsten Start angemerkt wird. In diesem Fall muss ein Start zunächst mit der repair-Option erfolgen. Um diese Probleme zu umgehen kann zum Experimentieren einfach eine Batch-Datei angelegt werden, in der zunächst immer erst die Reparatur ausgeführt wird. Diese Start-Datei hat dann folgenden Inhalt.

```
C:\mongodb\bin\mongod --dbpath C:\mongodb\daten --repair
C:\mongodb\bin\mongod --dbpath C:\mongodb\daten
```

17.3 Erste Nutzung

Zur ersten Nutzung wird wieder eine Windows PowerShell geöffnet. Da im Beispiel das bin-Verzeichnis nicht im Pfad eingetragen ist, wird wieder in das bin-Verzeichnis gesteuert und die Verbindung mit „.\mongo“ gestartet. Das Fenster wartet jetzt auf Eingaben zur Interaktion mit der Datenbank.

```
PS C:\mongodb\bin> .\mongo
MongoDB shell version: 3.2.9-rc0-5-ga5191eb
connecting to: test
Server has startup warnings:
2016-08-07T19:48:11.254+0200 I CONTROL  [initandlisten]
2016-08-07T19:48:11.254+0200 I CONTROL  [initandlisten] ** WARNING: This 32-bit MongoDB binary is depr
ecated
2016-08-07T19:48:11.254+0200 I CONTROL  [initandlisten] ** NOTE: This is a 32-bit MongoDB binary runni
ng on a 64-bit operating
2016-08-07T19:48:11.256+0200 I CONTROL  [initandlisten] **       system. Switch to a 64-bit build of Mo
ngoDB to
2016-08-07T19:48:11.257+0200 I CONTROL  [initandlisten] **       support larger databases.
2016-08-07T19:48:11.258+0200 I CONTROL  [initandlisten]
2016-08-07T19:48:11.259+0200 I CONTROL  [initandlisten]
2016-08-07T19:48:11.260+0200 I CONTROL  [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2016-08-07T19:48:11.261+0200 I CONTROL  [initandlisten] **       32 bit builds are limited to less tha
n 2GB of data (or less with --journal).
2016-08-07T19:48:11.262+0200 I CONTROL  [initandlisten] **       Note that journaling defaults to off
for 32 bit and is currently off.
2016-08-07T19:48:11.263+0200 I CONTROL  [initandlisten] **       See http://dochub.mongodb.org/core/32
bit
2016-08-07T19:48:11.264+0200 I CONTROL  [initandlisten]
>
```

Wenn man das eigentliche Fenster der Datenbank beobachtet, sieht man in der letzten Zeile, dass eine Verbindung akzeptiert wurde.

```
2016-08-07T19:52:56.334+0200 I NETWORK  [initandlisten] connection accepted from 127.0.0.1:64578 #2 (1 connection now open)
```

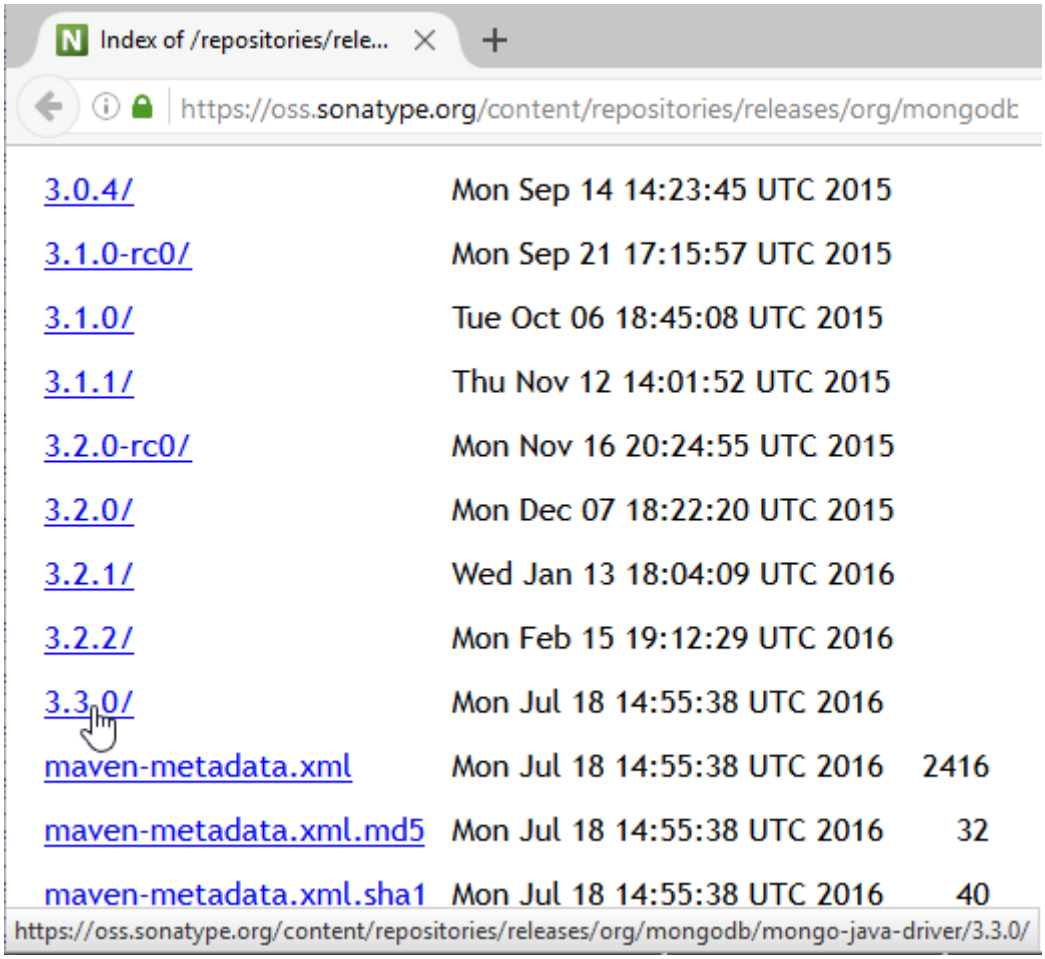
Eine Übersicht über die Datenbankbefehle gibt der Befehl „help“ aus.

```
> help
        db.help()                    help on db methods
        db.mycoll.help()             help on collection methods
        sh.help()                    sharding helpers
        rs.help()                    replica set helpers
        help admin                   administrative help
        help connect                 connecting to a db help
        help keys                    key shortcuts
        help misc                    misc things to know
        help mr                      mapreduce

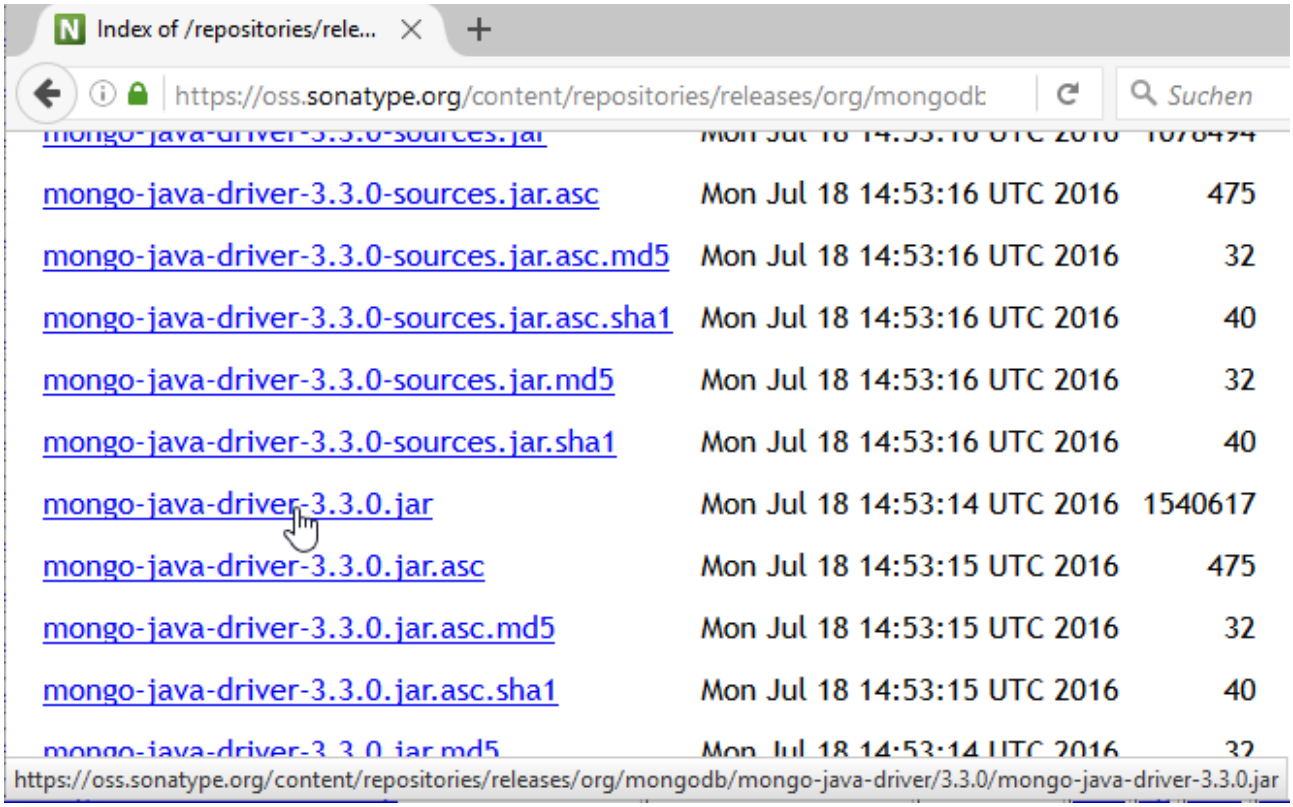
        show dbs                     show database names
        show collections             show collections in current database
        show users                   show users in current database
        show profile                 show most recent system.profile entries wit
h time >= 1ms
        show logs                    show the accessible logger names
        show log [name]              prints out the last segment of log in memor
y, 'global' is default
        use <db_name>                set current database
        db.foo.find()                list objects in collection foo
        db.foo.find( { a : 1 } )     list objects in foo where a == 1
        it                           result of the last line evaluated; use to f
urther iterate
        DBQuery.shellBatchSize = x   set default number of items to display on s
hell
        exit                         quit the mongo shell
>
```

17.4 Java und MongoDB

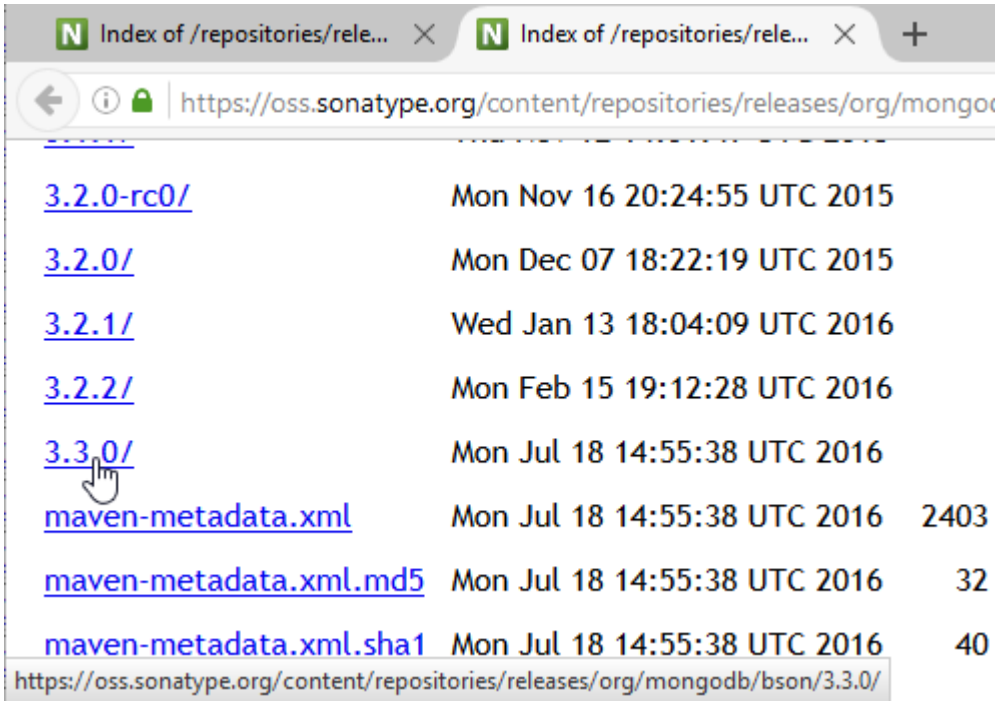
Vergleichbar zu anderen Datenbanken wird eine spezielle Bibliothek zum Aufbau der Verbindung und zur Nutzung der Datenbank benötigt. Der benötigte Java-Driver kann direkt von der Seite https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongo-java-driver/ geladen werden. Dazu wird in das Verzeichnis mit der höchsten Versionsnummer gewechselt, die kein „rc“ für „release candidate“ enthält. Im Beispiel ist es 3.3.0.



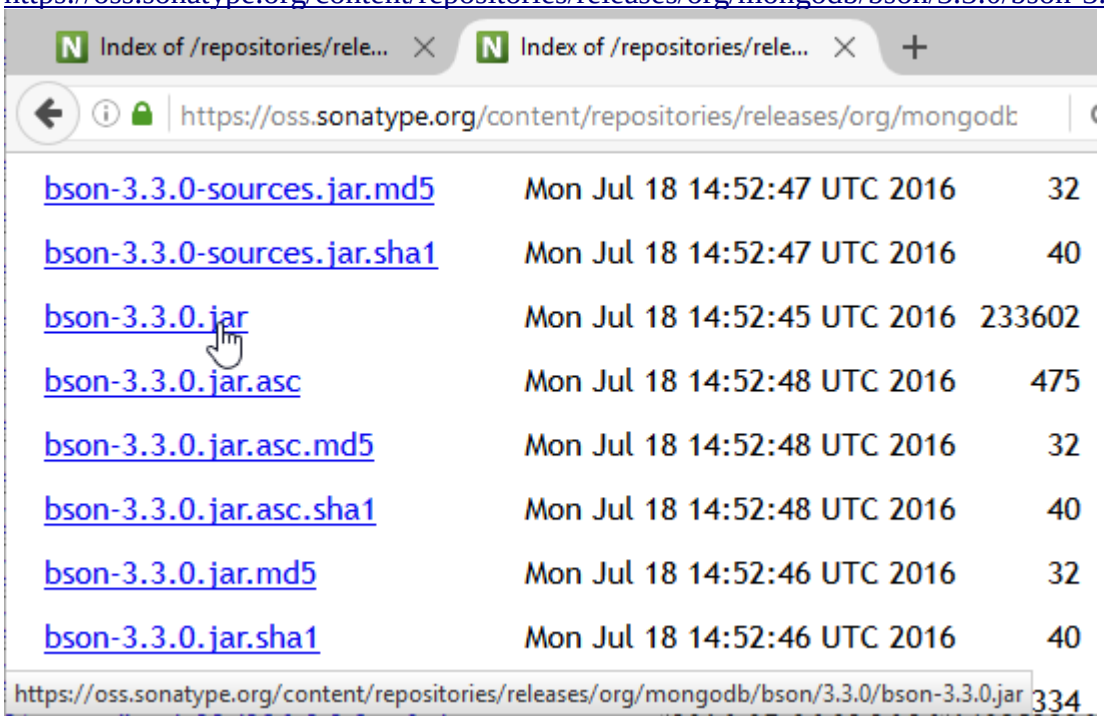
Aus dem Verzeichnis wird die Datei, mongodb-driver-3.3.0.jar heruntergeladen.



Weiterhin wird ein BSON-Support benötigt, der als Bibliothek bson-3.3.0.jar vorliegt und von der Seite https://oss.sonatype.org/content/repositories/releases/org/mongodb/bson/ heruntergeladen werden kann, https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongo-java-driver/3.3.0/mongo-java-driver-3.3.0.jar. Wieder wird dazu in die höchste stabile Versionsnummer gewechselt, die nicht identisch mit der des Java Driver sein muss.

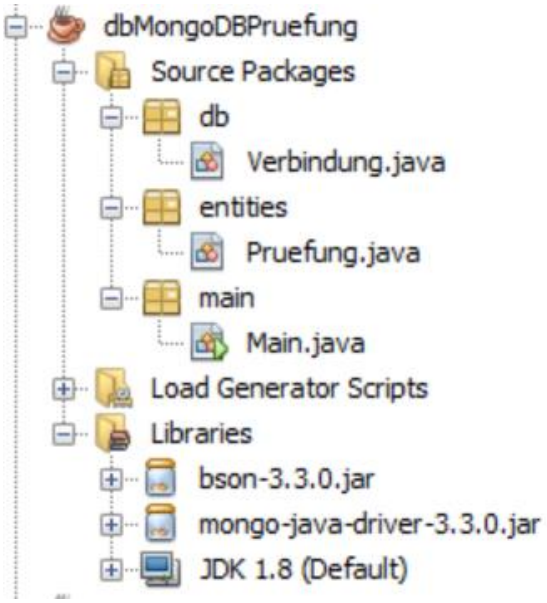


Es wird nur die eine Datei bson-3.3.0.jar benötigt, https://oss.sonatype.org/content/repositories/releases/org/mongodb/bson/3.3.0/bson-3.3.0.jar.



Die heruntergeladenen Jar-Dateien werden wie z. B. in „4.6 Nutzung von Bibliotheken/Jar-Files (NetBeans 8)“ beschrieben, in ein neues Java-Projekt eingebunden und danach kann die Nutzung

erfolgen.



Der nachfolgende Code zeigt den Standard-Verbindungsaufbau zu einer lokalen Datenbank, die den Namen „db01“ hat und zwei Collections „pruefungen“ und „zaehler“ benutzt. Die zweite Collection enthält nur ein Dokument, in der ein Zähler steht, der hochgezählt werden kann.

```
package db;

import com.mongodb.MongoClient;
import com.mongodb.MongoClientURI;
import com.mongodb.client.AggregateIterable;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MapReduceIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoIterable;
import com.mongodb.client.model.Filters;
import com.mongodb.client.model.Projections;
import com.mongodb.client.model.Sorts;
import entities.Pruefung;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import org.bson.Document;

public class Verbindung {

    private MongoClient mongoClient;
    private MongoDatabase db;
    private MongoCollection<Document> coll;
```

```
    private MongoCollection<Document> counter;

    public Verbindung() {
        MongoClientURI connectionString = new MongoClientURI("mongodb://localhost:27017");
        this.mongoClient = new MongoClient(connectionString);
        this.db = mongoClient.getDatabase("db01");
        this.coll = db.getCollection("pruefungen");
        this.counter = db.getCollection("zaehler");
        if (counter.count() == 0) {
            this.counter.insertOne(new Document("wert", 10));
            this.beispieldaten();
        }
    }

    public void schliessen() {
        this.mongoClient.close();
    }

    public void allesLoeschen() {
        this.coll.drop();
        this.counter.drop();
    }

    private void beispieldaten() {
        Pruefung[] pr = {new Pruefung(this.gibID(), 100, "DB", 1.0),
            new Pruefung(this.gibID(), 101, "DB", 3.0),
            new Pruefung(this.gibID(), 102, "DB", 5.0),
            new Pruefung(this.gibID(), 100, "Prog1", 2.3),
            new Pruefung(this.gibID(), 101, "Prog1", 3.7),
            new Pruefung(this.gibID(), 100, "Prog2", 1.7),
           // new Pruefung(this.gibID(), 100, "Prog3", 1.7), // neu
            new Pruefung(this.gibID(), 102, "Prog2", 2.0)
        };
        for (Pruefung p : pr) {
            this.hinzu(p);
        }
    }

    public void hinzu(Pruefung p) {
        this.coll.insertOne(p.asDocument());
    }


package entities;

import org.bson.Document;

public class Pruefung {
    private int _id;
```

```
private int mat;
private String fach; // eigentlich Modul-Identifikator
private double note; // besser int fuer Hundertstel

public Pruefung(){}

public Pruefung(int _id, int mat, String fach, double note) {
    this._id = _id;
    this.mat = mat;
    this.fach = fach;
    this.note = note;
}

public int get_id() {
    return _id;
}

public void set_id(int _id) {
    this._id = _id;
}

public int getMat() {
    return mat;
}

public void setMat(int mat) {
    this.mat = mat;
}

public String getFach() {
    return fach;
}

public void setFach(String fach) {
    this.fach = fach;
}

public double getNote() {
    return note;
}

public void setNote(double note) {
    this.note = note;
}

public Document asDocument(){
  Document erg = new Document()
          .append("_id", this._id)
          .append("mat", this.mat)
```

```
                .append("fach", this.fach)
                .append("note",this.note);
        return erg;
    }

    public static Pruefung asPruefung(Document doc){
        return new Pruefung(doc.getInteger("_id")
                            ,doc.getInteger("mat")
                            ,doc.getString("fach")
                            ,doc.getDouble("note"));
    }

    @Override
    public String toString() {
        return "Pruefung{" + "_id=" + _id + ", mat=" + mat + ", fach=" + fach
                            + ", note=" + note + '}';
    }
}
```

Zur Analyse und Bearbeitung der Daten werden viele Werkzeuge angeboten, die Installation und Nutzung eines solchen Clients ist im Anfang von Kapitel „15 Installation von PlugIns - MongoDB Client“ beschrieben.

18 UMLet

UMLet ist ein recht intuitiv zu bedienendes Werkzeug zur Erstellung von UML-Diagrammen. Dabei handelt es sich um ein Skizzenwerkzeug, das z. B. keine Code-Erzeugung oder Reverse-Engineering ermöglicht. Der wesentliche Vorteil neben der recht schnellen Erstellung, ist die Möglichkeit beliebige Diagrammelemente in einem Diagramm zu kombinieren, was gerade bei frühen Projektüberlegungen sinnvoll sein kann. UMLet gibt es als Eclipse-Plugin und als Standalone-Version, die hier betrachtet und später in NetBeans integriert wird.

18.1 Installation

Hinweis: Im Text wird die Version 13.3 von UMLet genutzt.

In der OOAD-Vorlesung wird der pragmatische Ansatz verfolgt, dass nicht alle Dokumente eines Projekts bei jeder Änderung aktualisiert werden müssen. Analysemodelle spielen am Anfang die zentrale Rolle, da sie zum Verständnis der Aufgabenstellung dienen und den Übergang zur Realisierung ermöglichen. Nach einer vollständigen Analyse, die bei inkrementell vorgehenden Projekten immer wieder ergänzt wird, werden die Analysedokumente und –modelle eingefroren und nicht mehr aktualisiert.

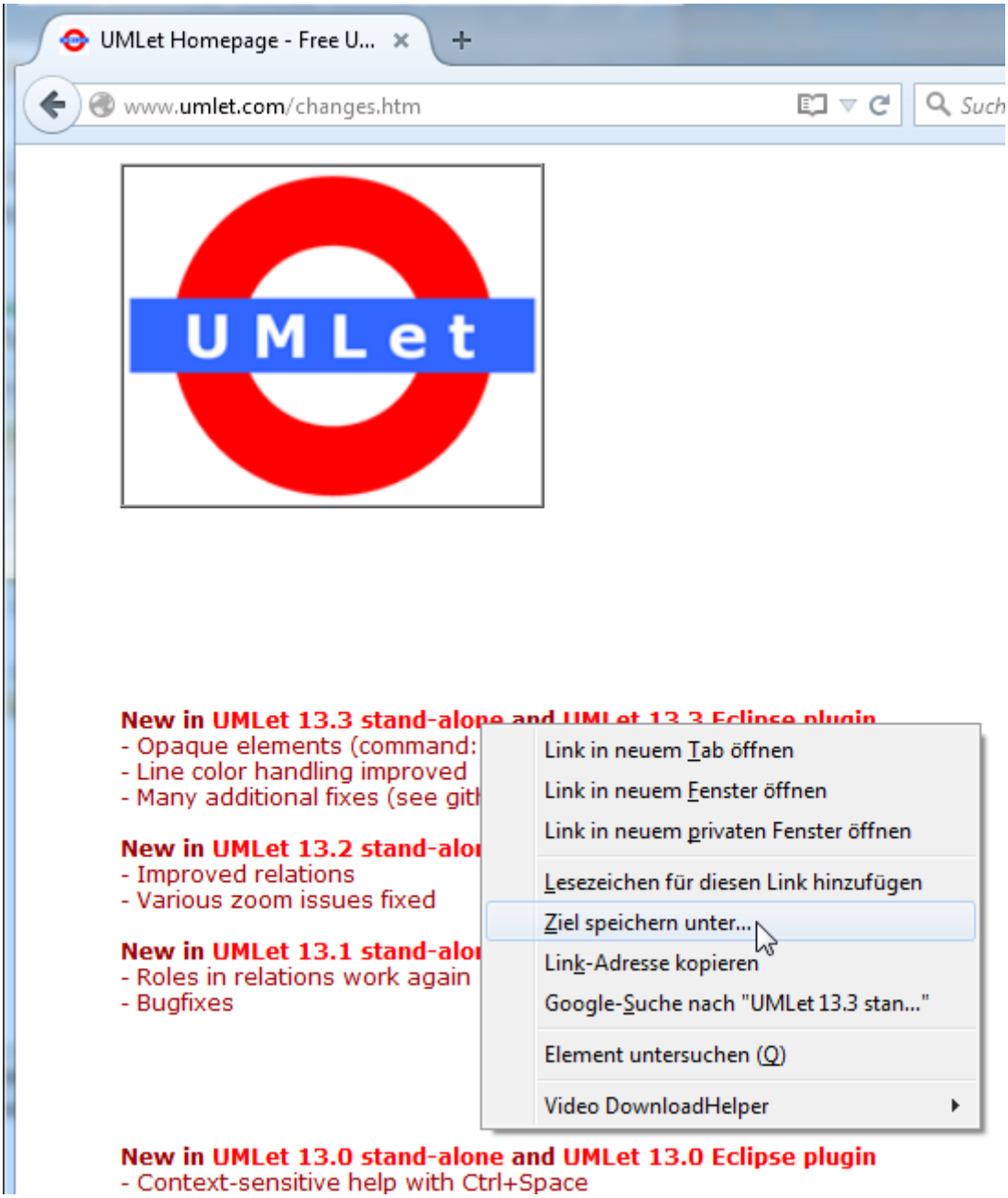
In der Veranstaltung wird das Werkzeug UMLet für die Erstellung von Skizzen genutzt. Das Werkzeug kann von der Web-Seite http://www.umlet.com/ geladen werden. Sollte es größere Probleme geben, ist die Stand-alone-Version zu nutzen. Weiterhin befindet sich ein Web-basiertes Werkzeug UMLetino http://www.umlet.com/umletino mit verwandter Funktionalität in Entwicklung. Bei der Nutzung ist darauf zu achten, dass erstellte Dateien wirklich als Dateien über den Browser abgespeichert werden, da der HTML5-Speicher nicht beliebig lang existieren muss.

Auf der Startseite wird der Download-Link genutzt.



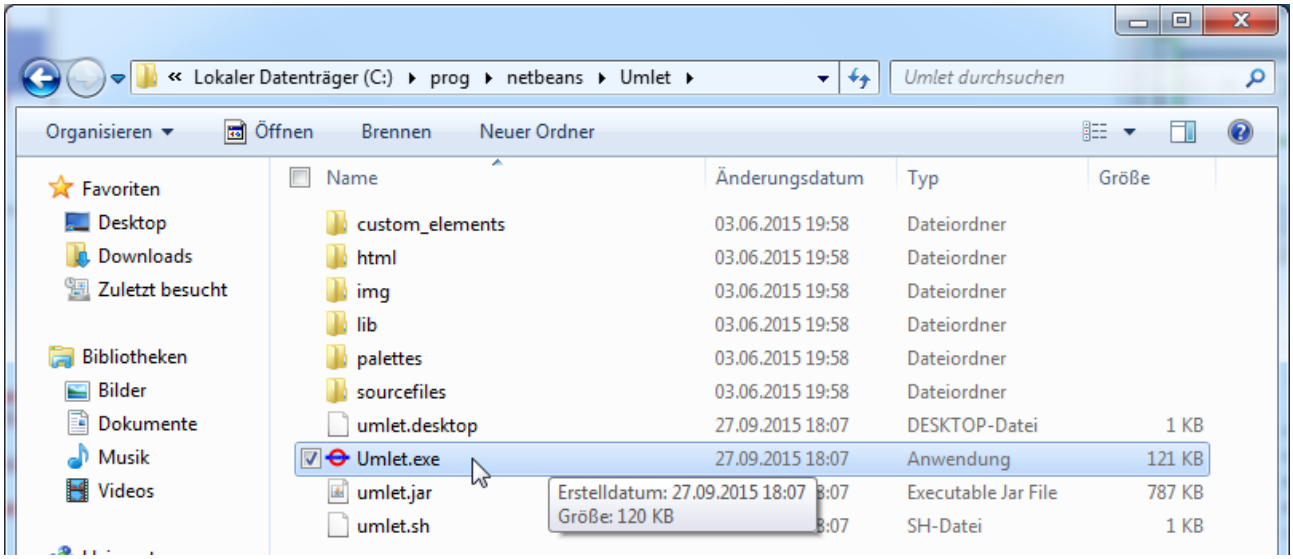
Die Download-Seite ist http://www.umlet.com/changes.htm.

Hier wird die aktuelle Stand-Alone-Version mit einem Rechtsklick heruntergeladen.

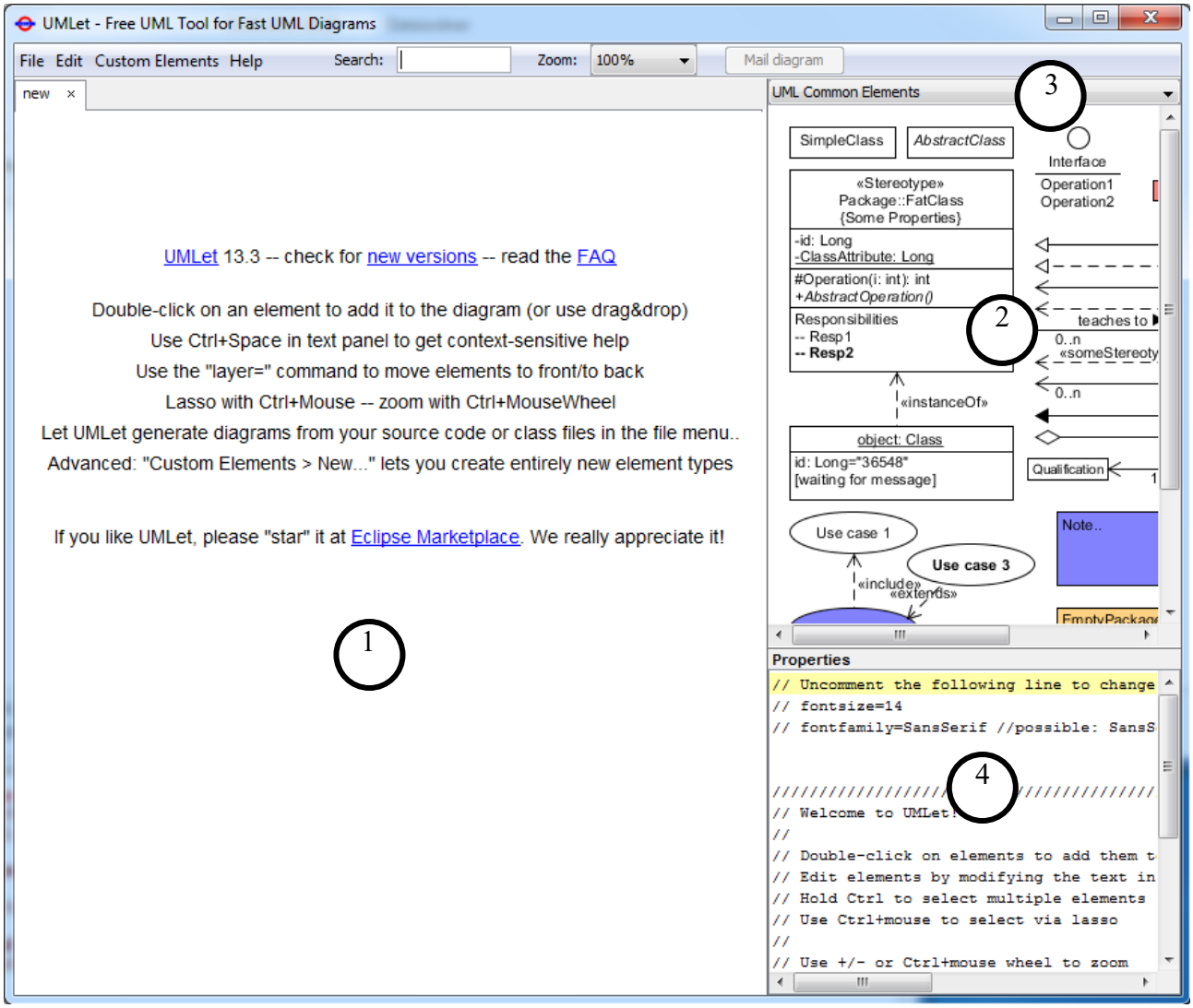


18.2 Erste Nutzung

Zum direkten Aufruf des Werkzeugs wird die zip-Datei ausgepackt und Umlet.exe mit einem Doppelklick gestartet. Alternativ kann in das Verzeichnis mit einer „Dos-Box“ (cmd-Werkzeug) oder unter anderen Betriebssystemen mit einem Konsolen-Fenster gegangen und das jar-File von Hand mit `java -jar umlet.jar` gestartet werden.



Da das Fenster relativ klein zum Arbeiten ist, sollte dieses Fenster vergrößert werden.

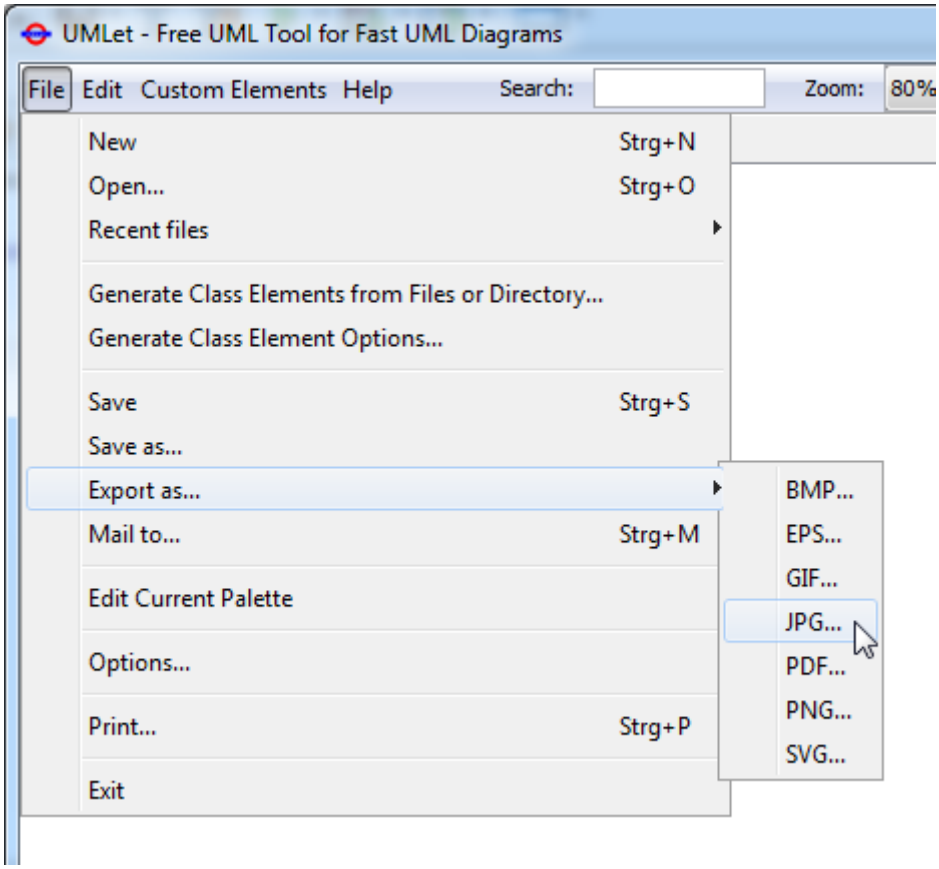


Im Werkzeug gibt es neben der üblichen Dateibehandlung die vier markierten wichtigen Bereiche.

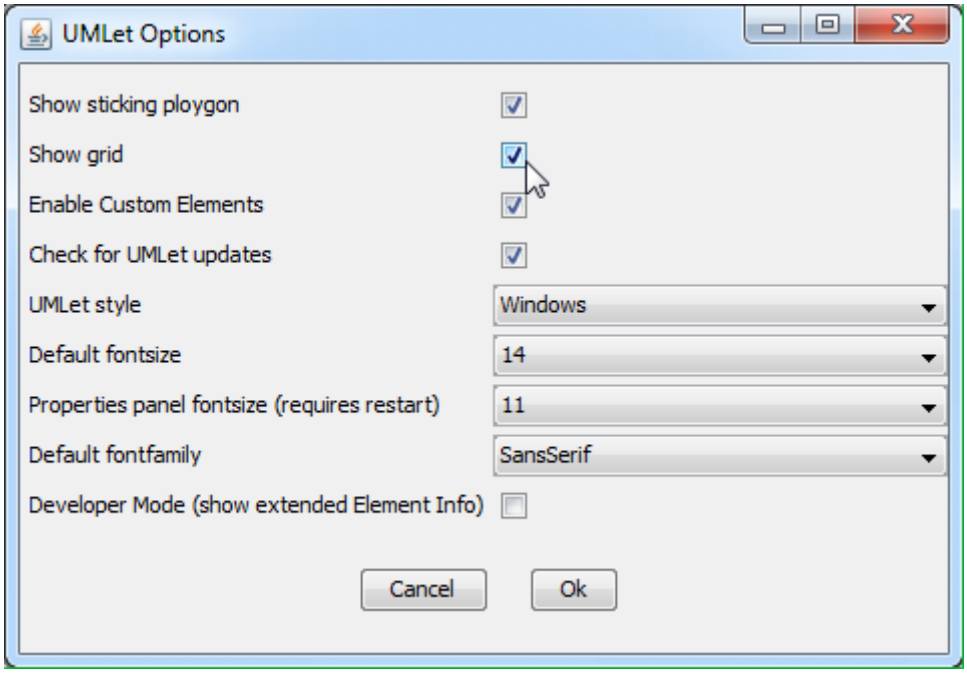
1. Dies ist der Zeichenbereich, in dem alle Elemente verknüpft und angezeigt werden.
2. In diesem Bereich wird eine Palette vom UML-Zeichenelementen angezeigt, die durch einen Doppelklick in den Zeichenbereich übernommen werden. (Diese Art der Steuerung ist etwas gewöhnungsbedürftig, aber recht effizient.)
3. In dieser Drop-Down-Box kann man unterschiedliche Paletten auswählen, die für unterschiedliche UML-Diagramme verschiedene Zeichenelemente anbieten.
4. Wenn man Beschriftungen z. B. von Aktivitäten oder Inhalte von Klassen ändern möchte, passiert dies immer in dieser Textbox.

Das File-Menü bietet die Möglichkeit die Datei unter einem eigenen Namen unter „Save As“ abzuspeichern. Weiterhin gibt es hier die Möglichkeit, das Diagramm in verschiedene graphische

Formate zu exportieren. Dabei ist es immer sinnvoll, das Diagramm möglichst genau links-oben im Editor zu platzieren.



Weiterhin gibt es hier das Options-Menü, bei dem u. a. Gitterlinien eingeblendet werden können, die bei der Ausrichtung von Elementen helfen.

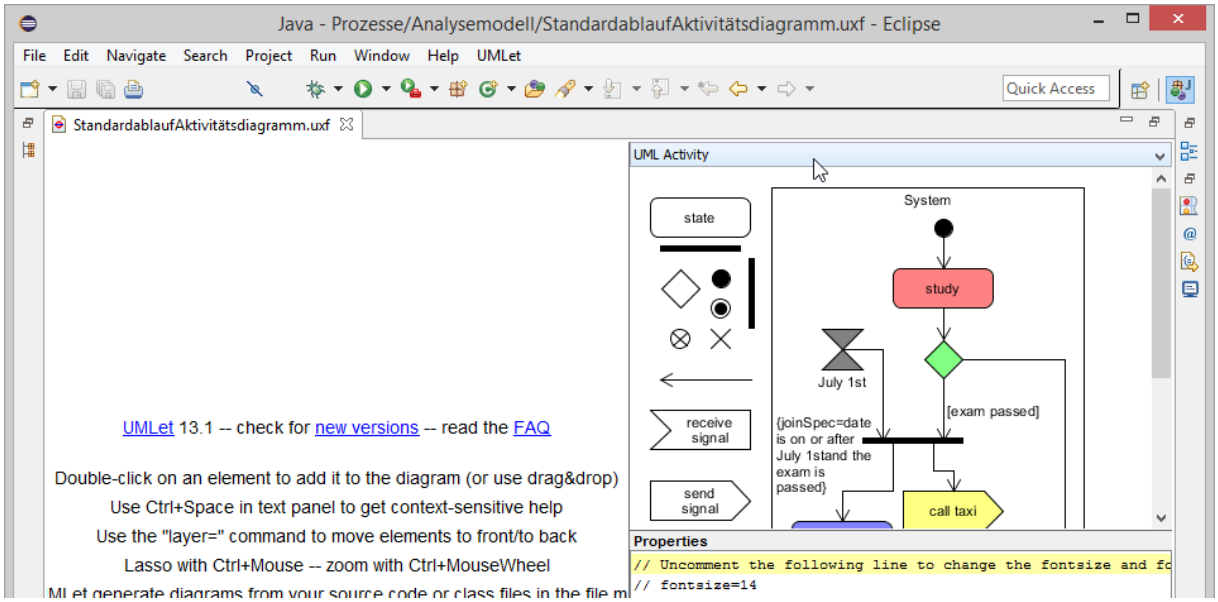


UMLet macht keine Syntaxprüfung, dass bedeutet, dass man beliebigen „Unsinn“ in die Diagramme zeichnen kann. Dies ist aber auch ein wichtiger Vorteil für die Analysephase, da man sich hier frei für eine Darstellungsform entscheidet. Wichtig ist nur, dass sie von jedem Projektbeteiligten gelesen werden kann und dass die Grundideen der Darstellungsweise dokumentiert sind.

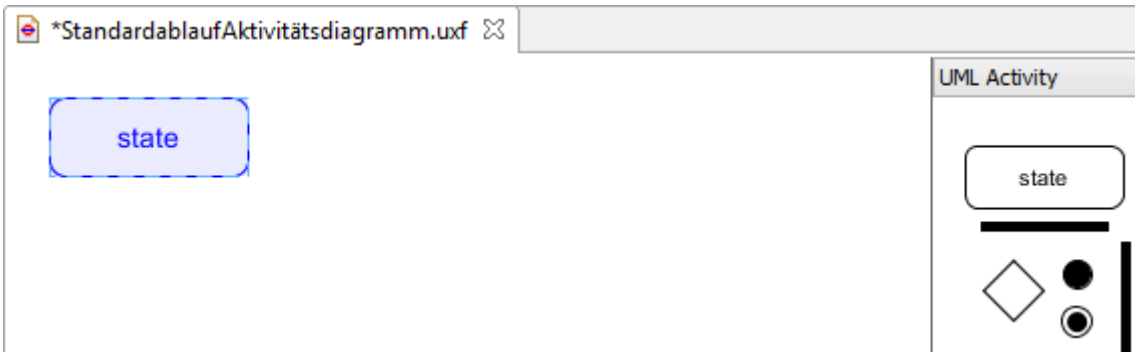
Die folgenden Anmerkungen gelten für die hier betrachtete Version und das Eclipse-Plugin. Einzelne Abbildungen stammen aus Eclipse, sind aber einfach auf die andere Version übertragbar.

Beim Arbeiten mit UMLet ist zu beachten, dass man mit einem Doppelklick auf ein Zeichenelement, egal ob rechts in der Palette oder links in der Zeichenfläche, das ausgewählte Element verdoppelt. Dies ist sehr hilfreich, wenn man z. B. mehrere Aktivitäten oder auch Pfeile im Zeichenbereich organisieren möchte. Bei Anfängern führt dieser Ansatz aber ab und zu, zu Problemen, so dass auch ein Drag-and-Drop von Elementen von der rechten Seite in das Zeichenfeld auf der linken Seite möglich ist. Da dabei allerdings auch Elemente der rechten Seite unabsichtlich verschoben oder gelöscht werden können, ist der Weg mit dem Doppelklick vorzuziehen.

Im Folgenden ist skizziert, wie man ein Aktivitätsdiagramm anlegen kann, dabei ist jeder Nutzer zum selbst Experimentieren aufgerufen. Zunächst wird die zu Aktivitätsdiagrammen gehörende Palette „UML Activity“ ausgewählt.

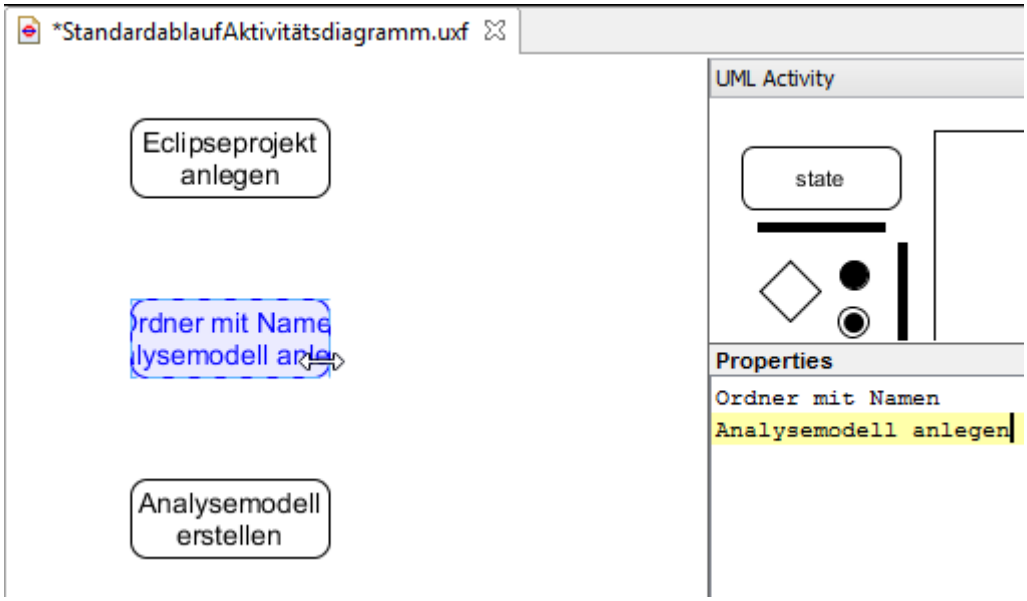


Danach wird auf der rechten Seite die Aktion, etwas unsauber als „state“ bezeichnet, doppelt angeklickt und so in den Zeichenbereich übernommen.

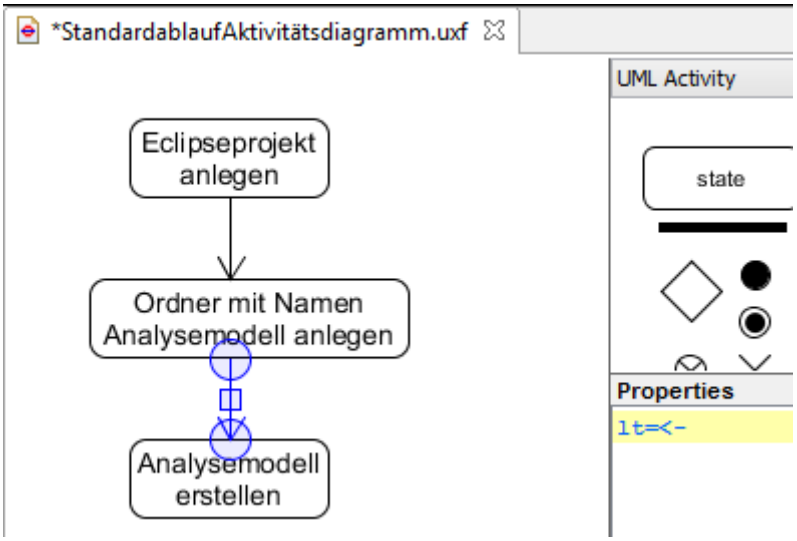


Um mehrere Aktionen zu erzeugen, wird dann ein Doppelklick auf der Aktion im Zeichenbereich ausgeführt. Graphische Elemente können generell mit gedrückter linker Maustaste verschoben und durch einen Doppelklick kopiert werden.

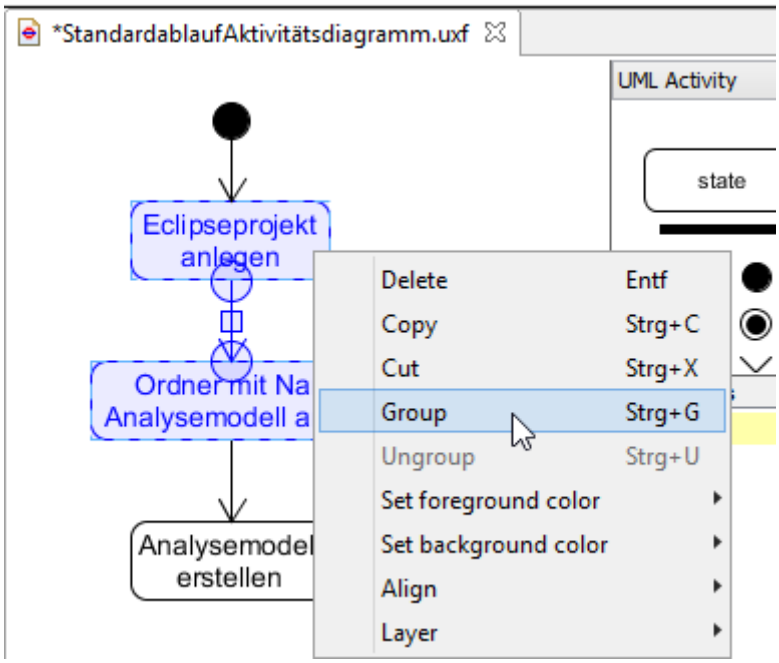
Die Beschriftung wird im erwähnten vierten Bereich geändert. Wenn man die Maus über ein Element schiebt, kann man dessen Größe anpassen.



Danach werden die anderen Elemente eingefügt und miteinander verbunden, Pfeile werden ebenfalls mit einem Doppelklick kopiert. Die mit Kreisen markierten Enden können über einen Linksklick verschoben werden. Das Quadrat in der Mitte des Pfeils ermöglicht die Verschiebung des gesamten Pfeils. Sitzen Pfeilende am Rand eines Objektes, werden später mit verschoben.

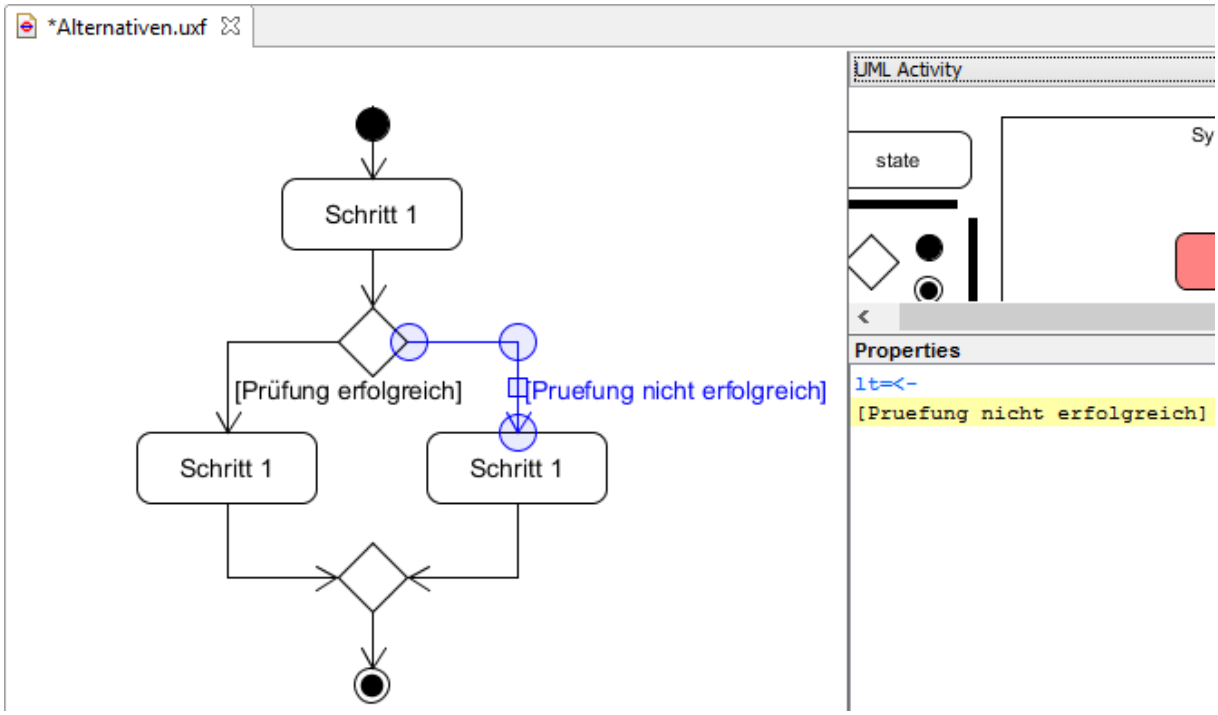


Klickt man neben das Diagramm und verschiebt bei gedrückter linker Maustaste die Maus, so wird das gesamte Diagramm verschoben. Man kann mit gedrückter Strg-Taste mehrere Elemente selektieren und diese dann mit einem Rechtsklick im dann sichtbaren Menü zu einer Gruppe zusammenfassen. Danach wird diese Gruppe immer zusammenbehandelt, was z. B. für das Verschieben einer Teilmenge der Diagrammelemente sehr hilfreich ist.



Nachdem ein Modell vollständig eingegeben wurde, muss es mit dem Punkt „Save“ im File-Menü gespeichert werden.

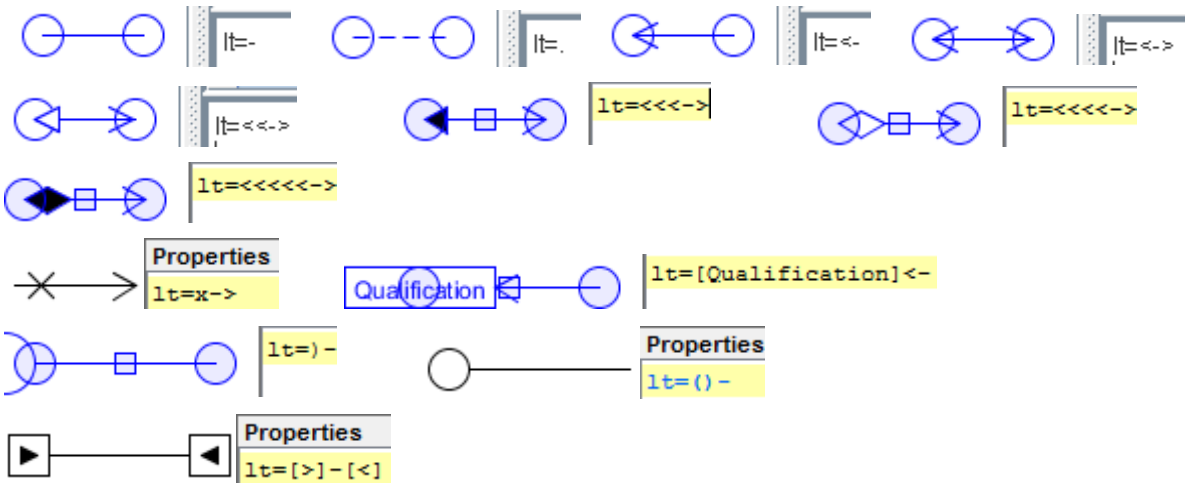
Möchte man Kanten z. B. bei Entscheidungen beschriften, ist zu beachten, dass die erste Zeile in der Beschriftungszeile gleich bleibt, da mit ihr angegeben wird, an welchen Enden Pfeilspitzen stehen sollen. Die folgende Abbildung zeigt links eine selektierte Kante und rechts das Textfeld, in dem die Beschriftung eingetragen wird. Knicke in Kanten erzeugt man, wenn man auf einer selektierten Kante an dem Punkt, an dem ein neuer Knickpunkt entstehen soll, an der Kante mit gedrückter linker Maustaste zieht. Der markierte Mittelpunkt der Kante dient dazu, die gesamte Kante zu verschieben und kann nicht zur Knickerstellung genutzt werden.



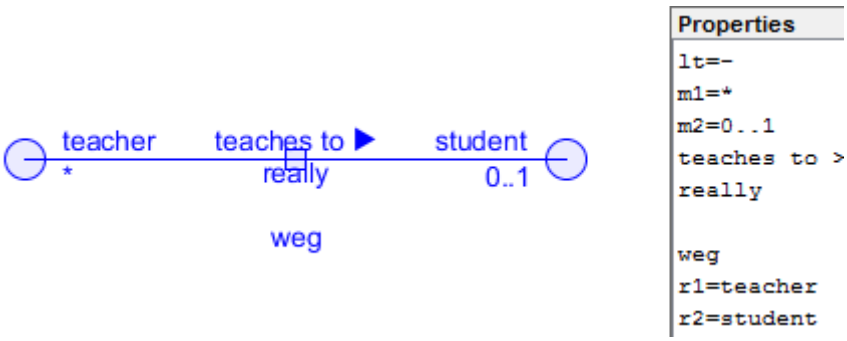
Bei der Bearbeitung von Klassen ist noch zu beachten, dass bei Texteingaben eine leere Zeile mit zwei Minuszeichen dazu führt, dass im Klassendiagramm ein Strich gemalt wird. Möchte man Leerzeilen einfügen, müssen diese mindestens ein Leerzeichen enthalten. Klassenvariablen und Klassenmethoden werden in der UML unterstrichen, dies ist durch das Voranstellen und Abschließen mit einem Unterstrich möglich. Es gibt einige weitere Steuerungsbefehle, von denen einige im Folgenden betrachtet werden.



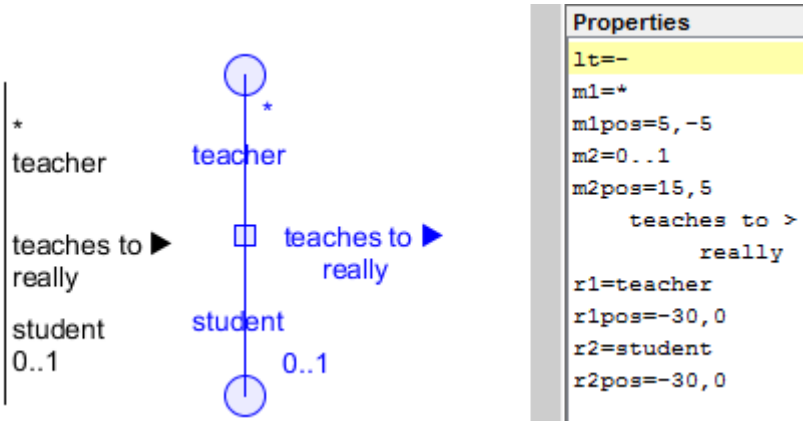
Beschäftigt man sich etwas genauer mit UMLet, gibt es noch einige wenige Steuerbefehle, die in der Beschriftungsbox eingegeben werden können. Die folgenden Bilder zeigen zunächst links die Linienart und rechts den eingegebenen Text, die Kreise zeigen nur, dass die Linie selektiert wurde.



Weiterhin gibt es verschiedene Plätze, an denen Eigenschaften an Linien notiert werden können.



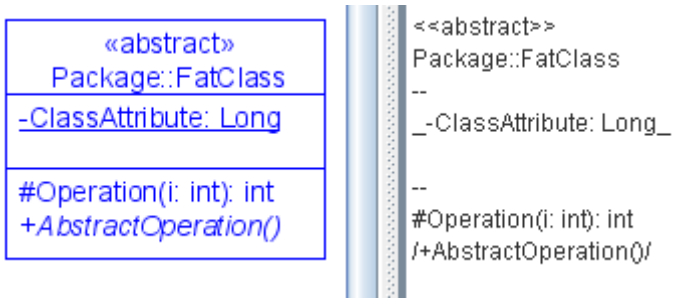
Da die Platzierung nicht immer optimal ist, kann man die Texte mit einzelnen Leerzeichen oder der Angabe von Pixelverschiebungen platzieren.



Ausgefüllte Elemente können eine Farbe haben.

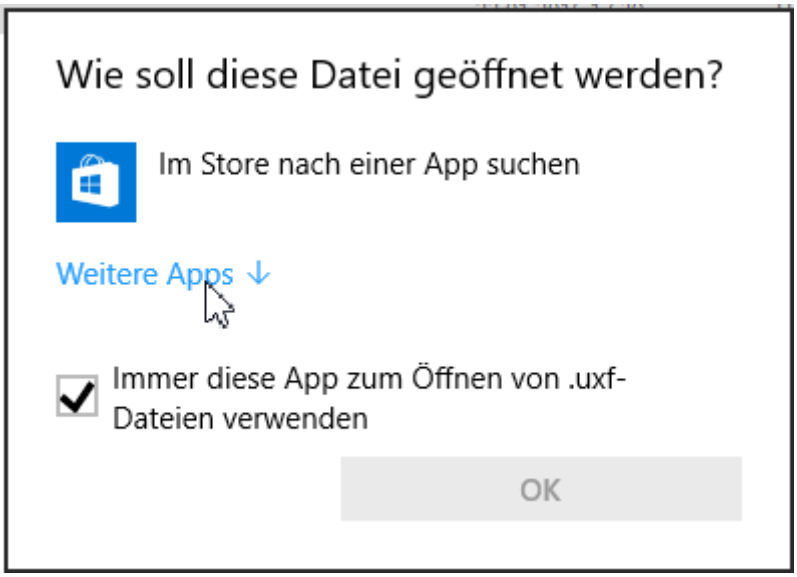


Für abstrakte Methoden müssen Schrägstriche um den Namen herum stehen, damit dieser kursiv dargestellt wird. Beo Klassenmethoden und Klassenvariablen steht unmittelbar vor und nach dem Namen ein Unterstrich.

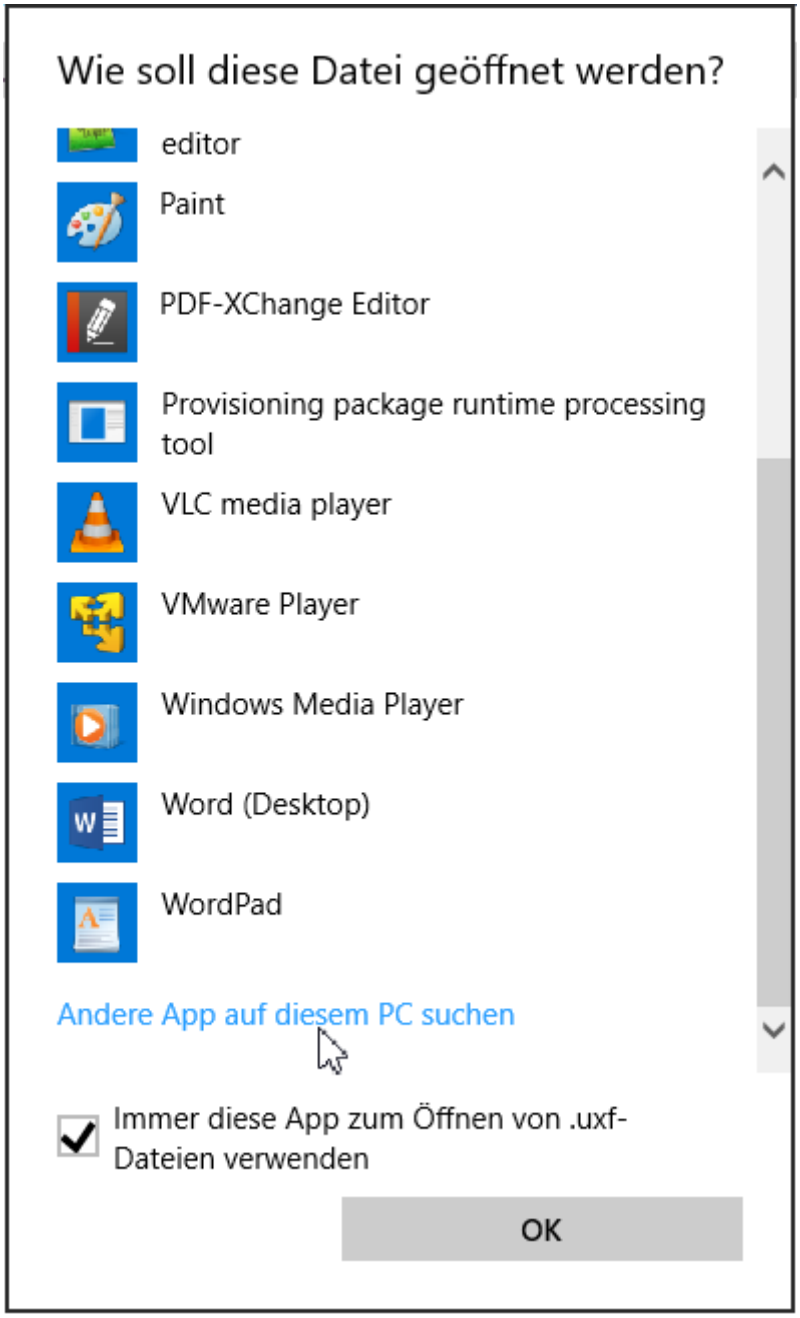


18.3 Verknüpfung von uxf-Dateien mit UMLet

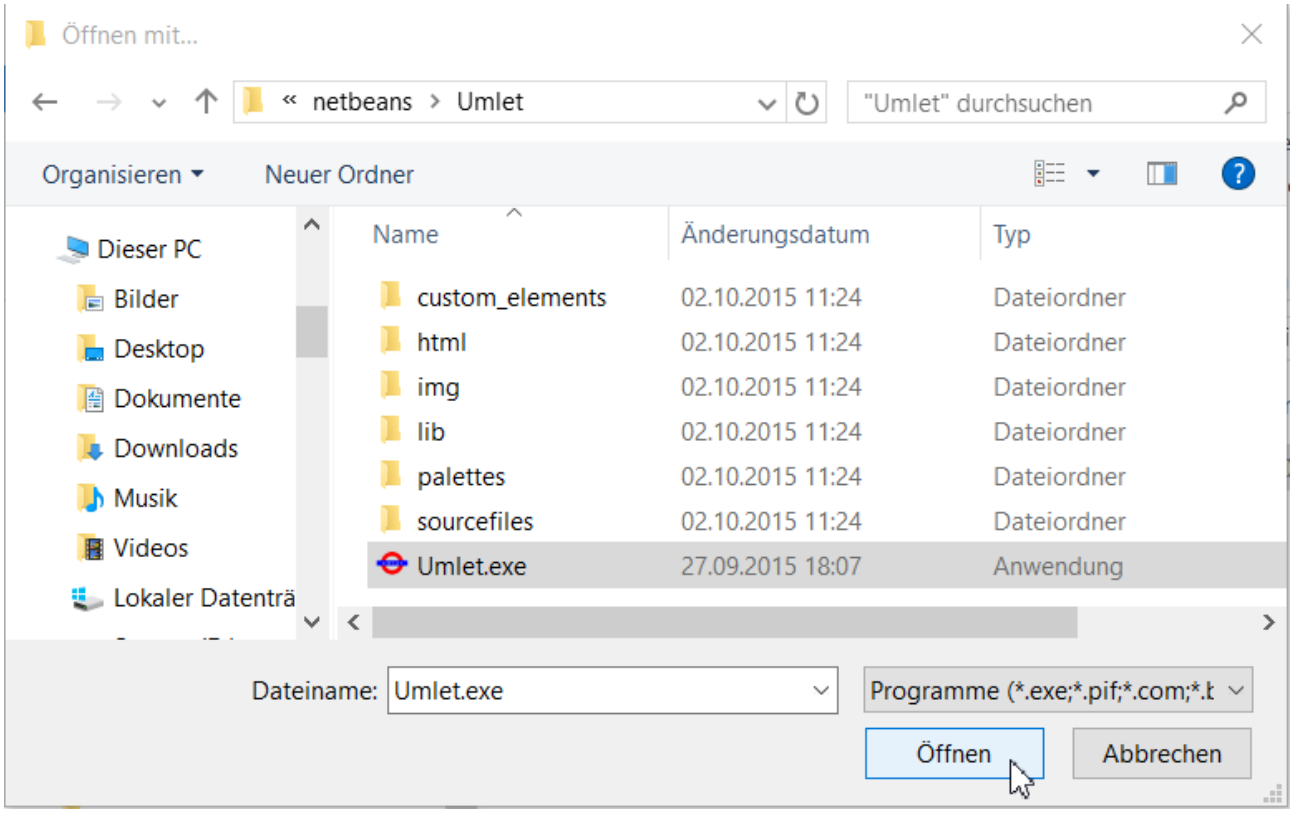
UMLet ist alleine nutzbar und hat keine echte Integration in NetBeans. Hier wird gezeigt, wie eine Verknüpfung mit den zugehörigen .uxf-Dateien und dadurch eine Integration in NetBeans erfolgt. Nach einem Doppelkick auf einer uxf-Datei öffnet sich die Frage nach dem zu nutzenden Programm. Es wird auf „Weiter Apps“ geklickt.



Es wird „Andere App auf diesem PC suchen“ geklickt.



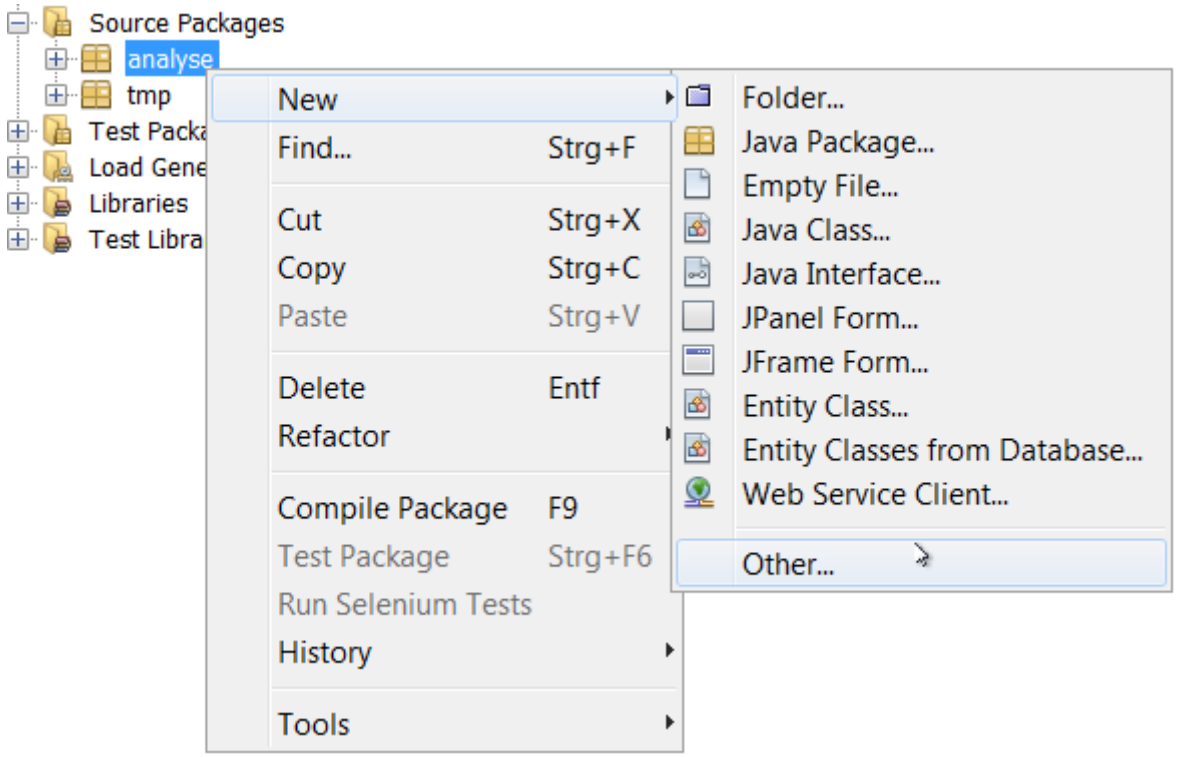
Nun wird zur ausführbaren Datei umlet.exe gesteuert, diese ausgewählt und „Öffnen“ geklickt.



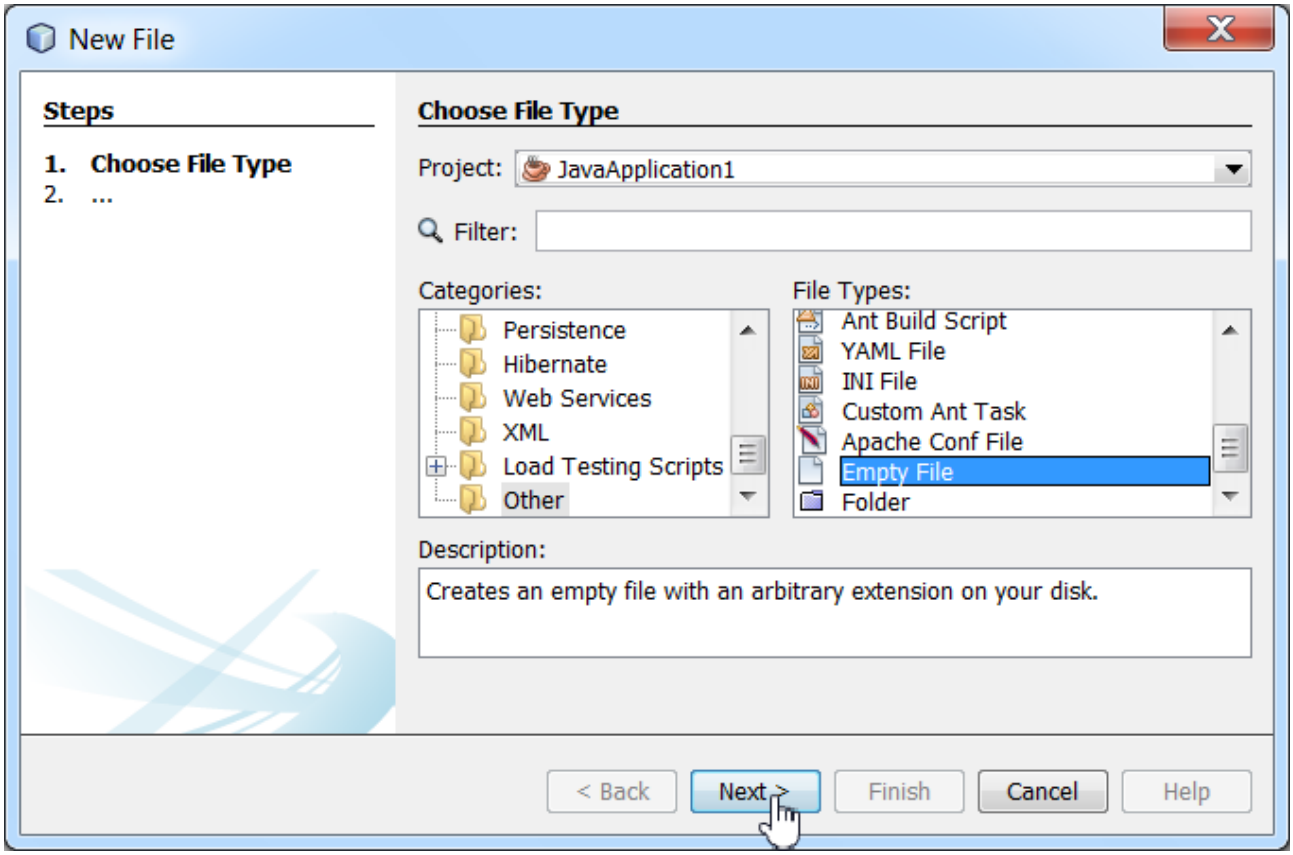
Danach öffnet sich UMLet und die Datei kann bearbeitet werden.

18.4 Anlegen einer UMLet-Datei in NetBeans

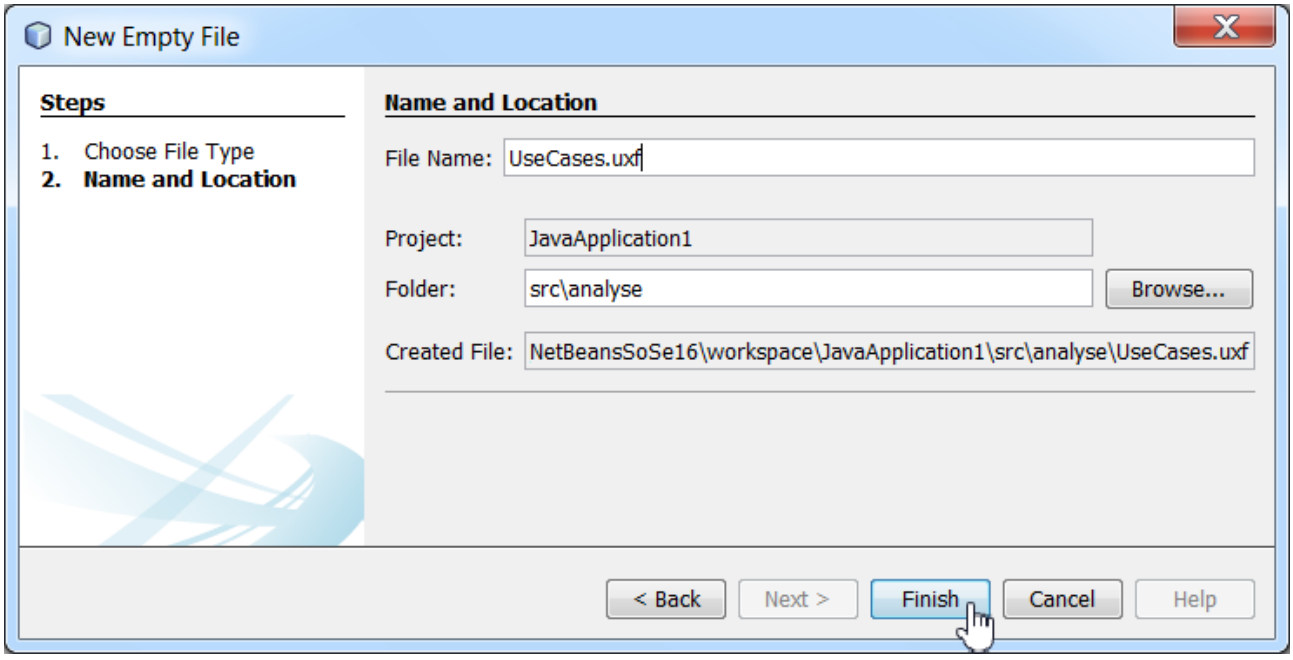
Generell kennt NetBeans UMLet nicht, berücksichtigt aber Dateiverknüpfungen des Systems. Um eine neue Datei in NetBeans zur Bearbeitung mit UMLet anzulegen, ist ein kleiner Trick notwendig. Es wird einfach eine „beliebige“ Datei mit Endung uxf angelegt. Ein sinnvoller Ansatz dazu ist es, zunächst ein Paket analyse im Projekt zu erzeugen. Danach wird ein Rechtsklick auf dem Ordner, hier „analyse“, gemacht und „New -> Other…“ gewählt.



Rechts wird „Other“ und links dann fast ganz unten „Empty File“ gewählt und „Next >“ geklickt.

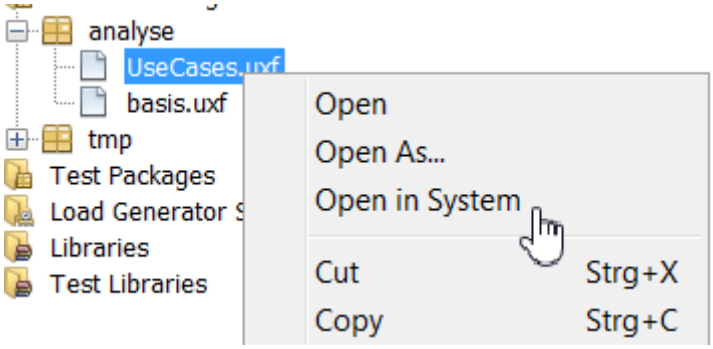


Nun wird oben der Dateiname eingegeben, auf die einzugebende Endung „.uxf“ geachtet und „Finish“ zum Abschluss geklickt.



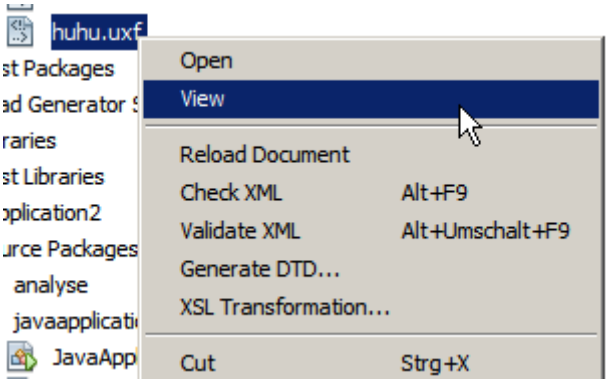
18.5 Öffnen einer UMLet-Datei in NetBeans

Wenn die Verknüpfung von UMLet mit uxf-Dateien erfolgt ist, kann in NetBeans ein Rechtsklick auf einer solchen Datei gemacht werden und es erscheint die Möglichkeit „Open in System“, die dann ausgewählt wird. UMLet öffnet sich und die Bearbeitung ist möglich. Es ist zu beachten, dass die Änderungen in UMLet gespeichert werden müssen.



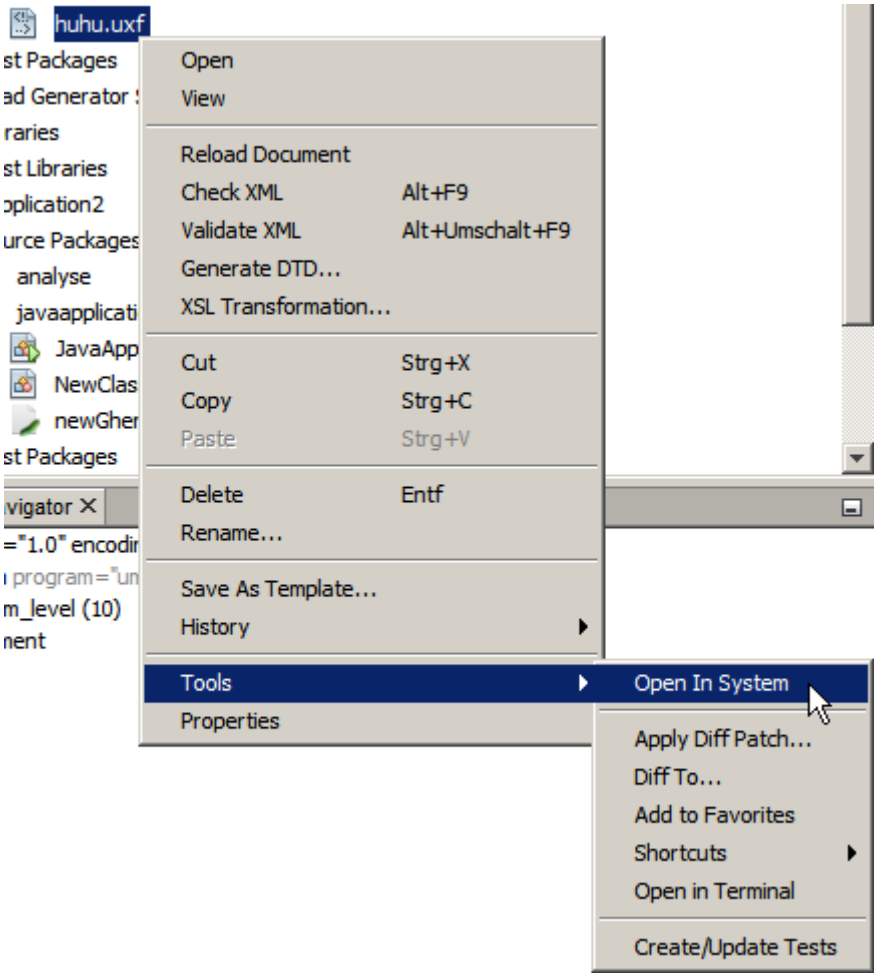
18.6 Öffnen einer UMLet-Datei in NetBeans bei fehlendem Menü-Punkt

Anscheinend abhängig von der NetBeans-Version, wird das vorher beschriebene Angebot „Open in System“ trotz vorhandener Verknüpfung im System nicht angeboten. Der Grund liegt anscheinend darin, dass es sich bei uxf-Dateien um Dateien im normalen XML-Format handelt, die von NetBeans besonders behandelt werden.



Hier ist es sinnvoll eine Erweiterung „Open In System“ manuell zu installieren, wie es unter „“ beschrieben ist. Der Download befindet sich unter http://plugins.netbeans.org/plugin/4830/open-in-system-jdk-1-6.

Nach der Installation wird nach einem Rechtsklick auf der Datei unten unter „Tools“ der Punkt „Open in System“ angeboten, der dann UMLet startet.

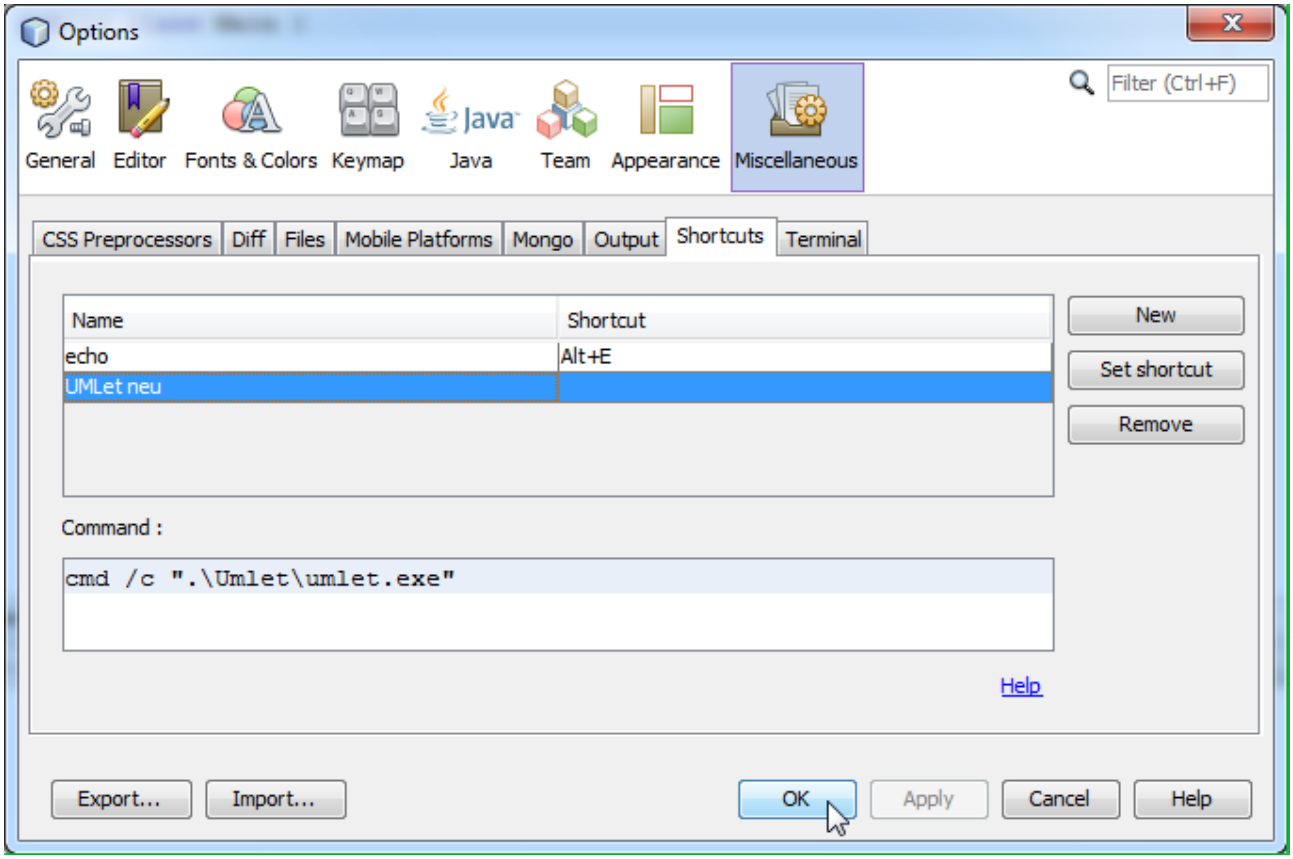


18.7 Einbindung von UMLet in NetBeans mit Schaltfläche

UMLet ist alleine nutzbar und hat keine echte Integration in NetBeans. Hier wird gezeigt, wie UMLet aus NetBeans heraus unter Nutzung eines Pluginsaufgerufen werden kann. Dazu wird das Plugin „NetBeans Command ShortCut“ genutzt, dessen Installation und erste Nutzung in „15.4 Installation eines heruntergeladenen Plugins - NetBeans Command Shortcuts“ beschrieben ist.

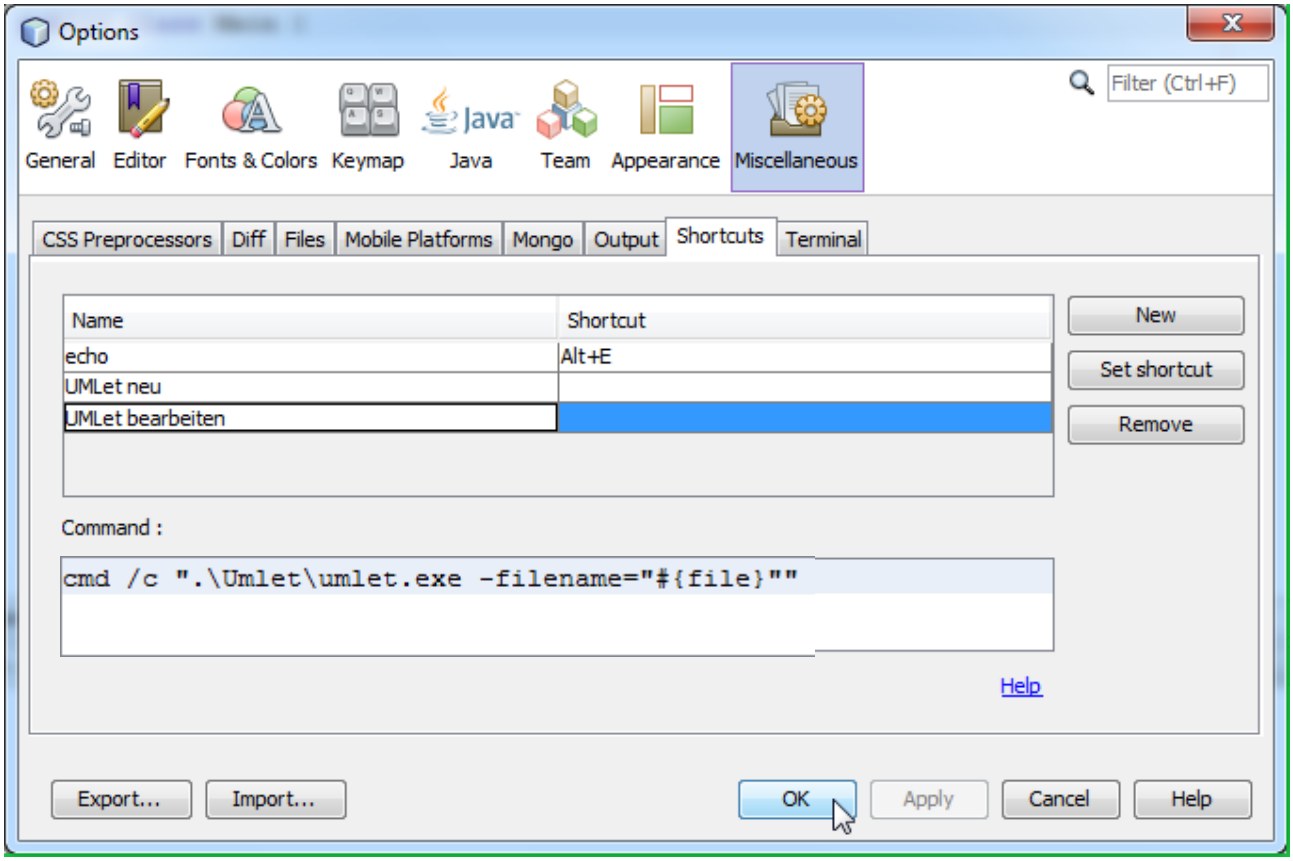
Es werden drei Shortcuts angelegt, der eine zum Starten von UMLet zur Erstellung einer neuen Datei, der andere zum Bearbeiten existierender UMLet-Dateien und der dritte zum Erstellen eines UMLet-Diagramms in einem bestimmten Ordner. Der Pfad zur ausführbaren UMLet-Datei umlet.exe muss dabei angepasst werden. Im Beispiel befindet sich UMLet in einem Verzeichnis UMLet, das sich direkt im NetBeans-Ordner befindet, was für ein portables NetBeans sinnvoll sein kann.
Der Befehl für den Start von UMLet sieht wie folgt aus.

```
cmd /c ".\Umlet\umlet.exe"
```



Der Befehl zur Bearbeitung einer Datei mit UMlet sieht wie folgt aus.

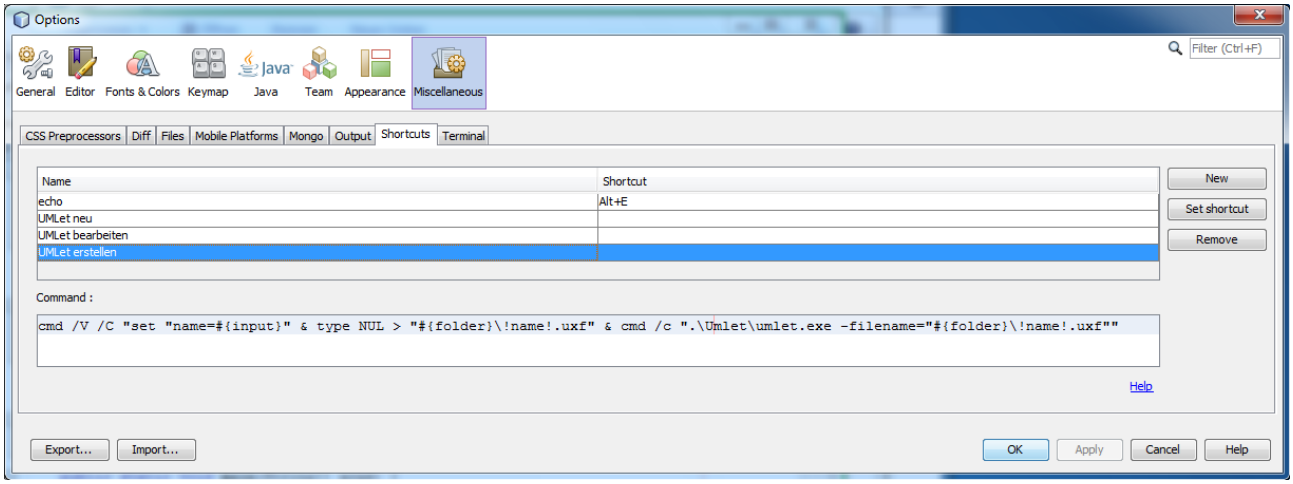
```
cmd /c ".\Umlet\umlet.exe -filename="#{file}""
```



Hinweis: Falls das Abspeichern des Shortcuts nicht funktioniert, muss nach der Eingabe in der Tabelle zunächst ein anderer Shortcut angeklickt und dann “Ok” gewählt werden.Alternativ kann es auch helfen, einmal auf ein „Shortcut“-Feld zu klicken, auch wenn nichts eingetragen wird. Der „Apply“-Button sollte auswählbar sein.

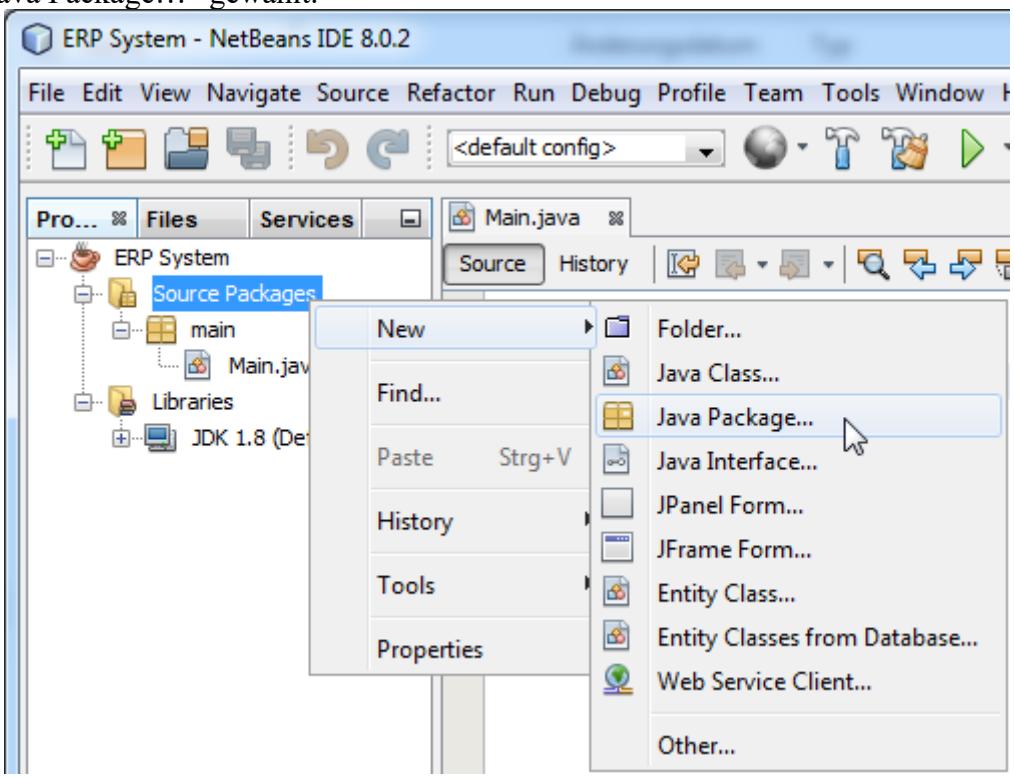
Der Befehl zum Erstellen eines Diagramms in einem bestimmten Ordner sieht wie folgt aus. In einer lokalen Systemvariablen wird der einzugebende neue Name gemerkt, diese Datei ohne Inhalt angelegt und dann geöffnet. Dies funktioniert nur, wenn im Verzeichnisnamen, also u. a. im Projektnamen, keine Leerzeichen vorkommen. Weiterhin kann der einmal gestartete Befehl nicht mehr sinnvoll abgebrochen werden.

```
cmd /V /C "set "name=#{input}" & type NUL > "#{folder}\!name!.uxf" & cmd /c ".\Umlet\umlet.exe -filename="#{folder}\!name!.uxf""
```

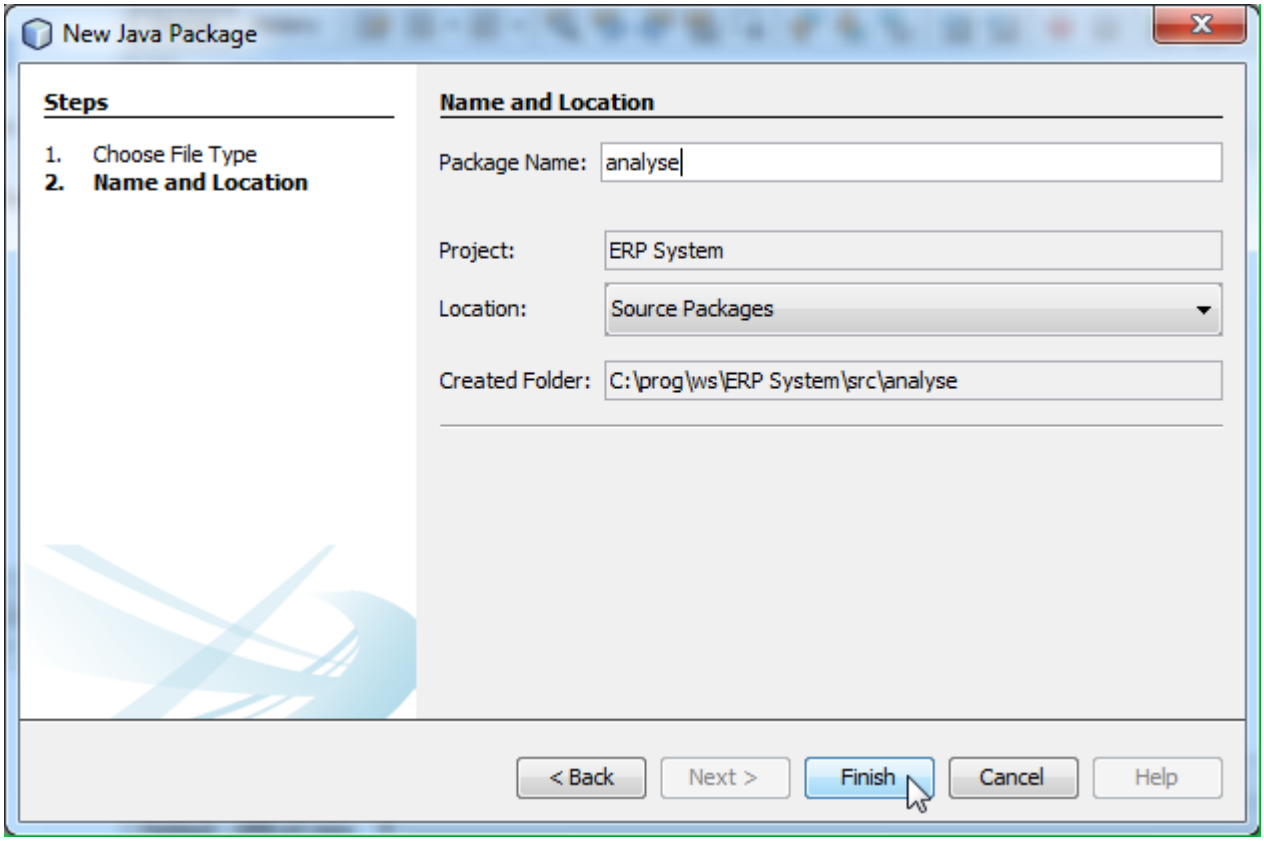


18.8 Nutzung von UMLet in NetBeans

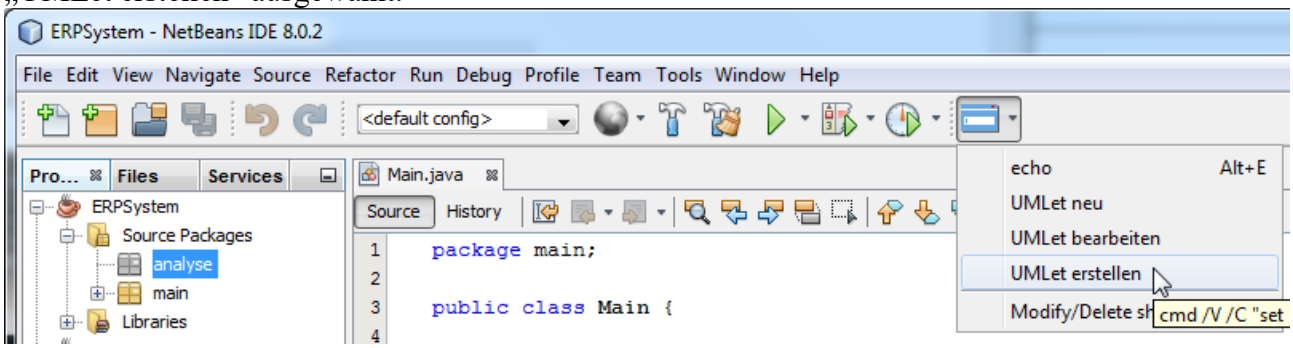
Um die UMLet-Diagramme in NetBeans zu integrieren wird ein kleiner unsauberer Trick genutzt, bei dem die Diagramme in einem speziellen Paket „analyse“ abgespeichert werden. Der Ansatz ist deshalb unsauber, da solche Pakete eigentlich nur für Code genutzt werden sollen.
Zur Erstellung des Pakets wird ein Rechtsklick auf dem Ordner „Source Packages“ gemacht und „New -> Java Package…“ gewählt.



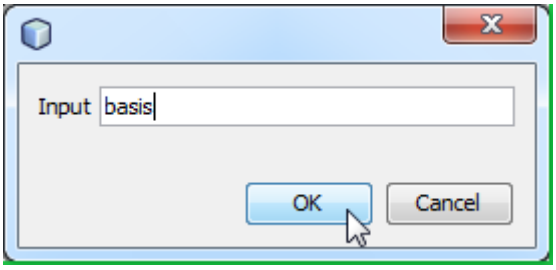
Der Name „analyse“ wird eingegeben und die Erstellung mit „Finish“ abgeschlossen. Die letzte Zeile zeigt, dass ein dazu passender Ordner angelegt wird.



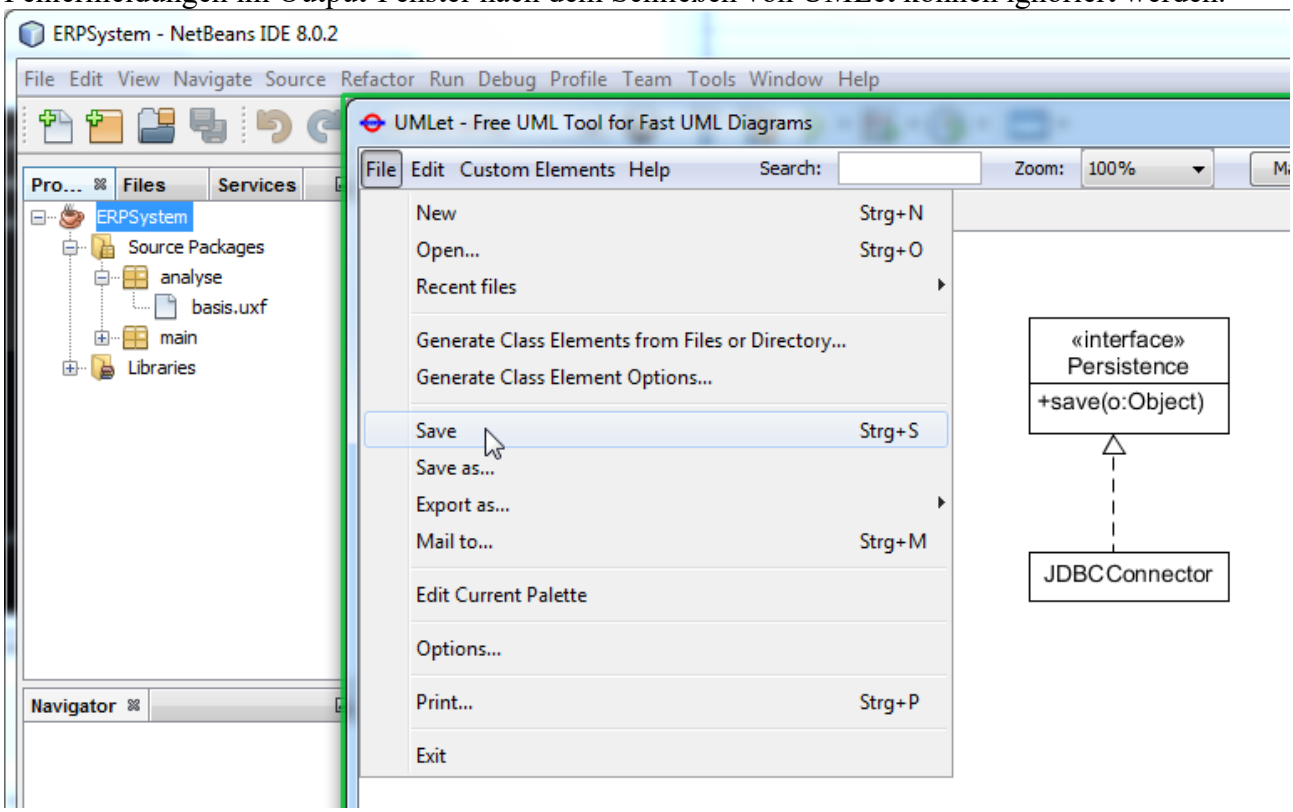
Zur Erstellung eines Diagramms wird der Ordner „analyse“ angeklickt und dann der Shortcut „UMLet erstellen“ ausgewählt.



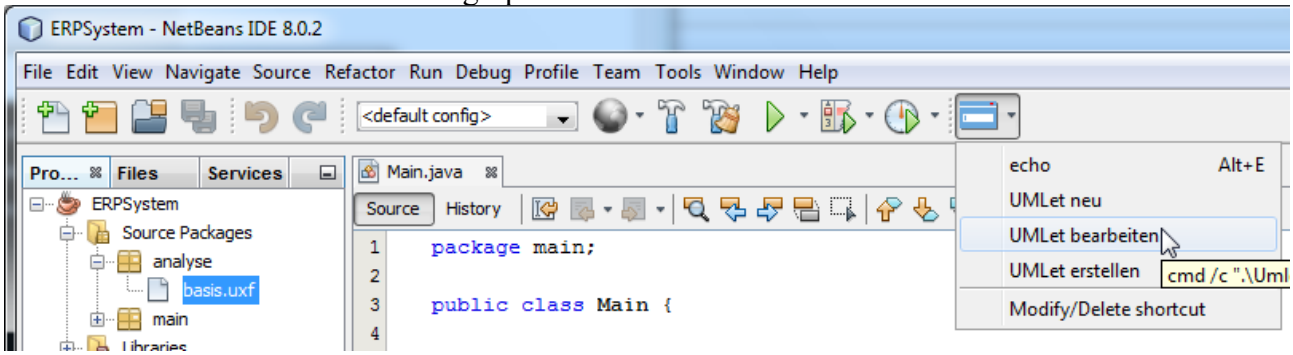
Es öffnet sich ein Fenster, in dem der Diagrammname (ohne Leerzeichen und Endung .uxf“) eingegeben wird.



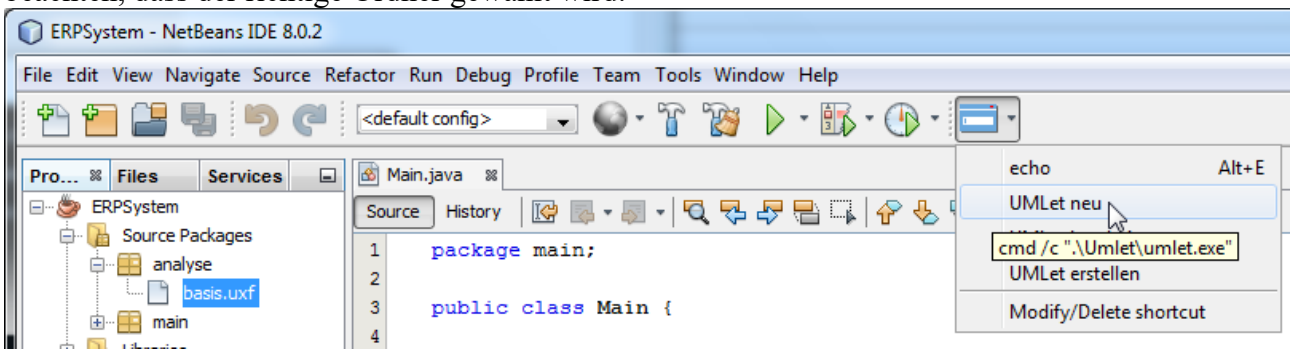
UMLet öffnet sich mit dem Dateinamen, weiterhin erkennt man in NetBeans, dass die Datei angelegt wird. Nach der Diagrammerstellung wird das Diagramm normal über UMLet gespeichert. Etwaige Fehlermeldungen im Output-Fenster nach dem Schließen von UMLet können ignoriert werden.



Um ein Diagramm zu öffnen, wird es angeklickt und der Shortcut „UMLet bearbeiten“ ausgewählt. Am Ende wird wieder über UMLet gespeichert.



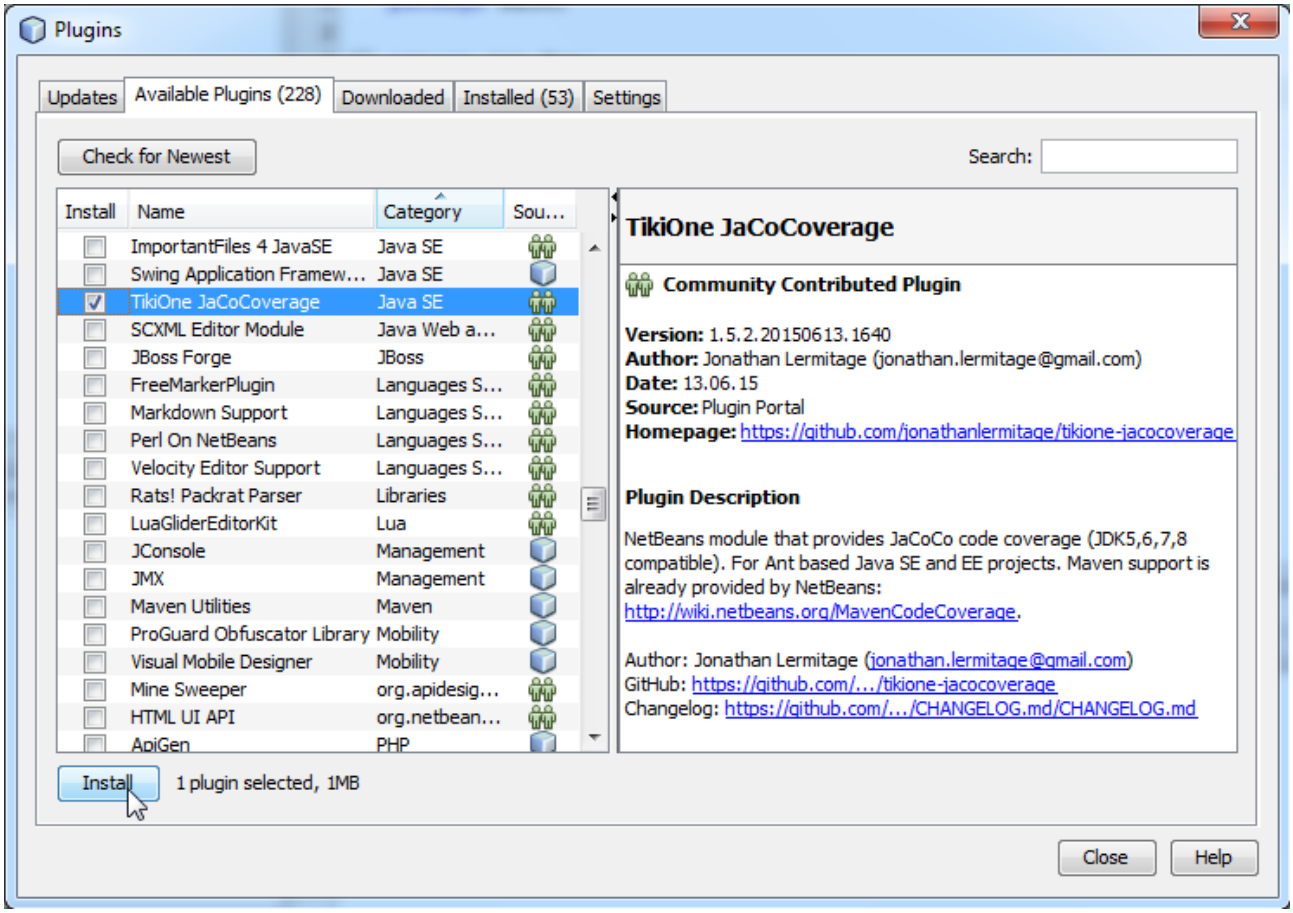
Falls es Probleme gibt oder Leerzeichen unbedingt genutzt werden sollen, kann der erste Shortcut „UMLet neu“ genutzt werden, der einfach UMLet startet. Beim Speichern ist dann mit „Save As“ zu beachten, dass der richtige Ordner gewählt wird.



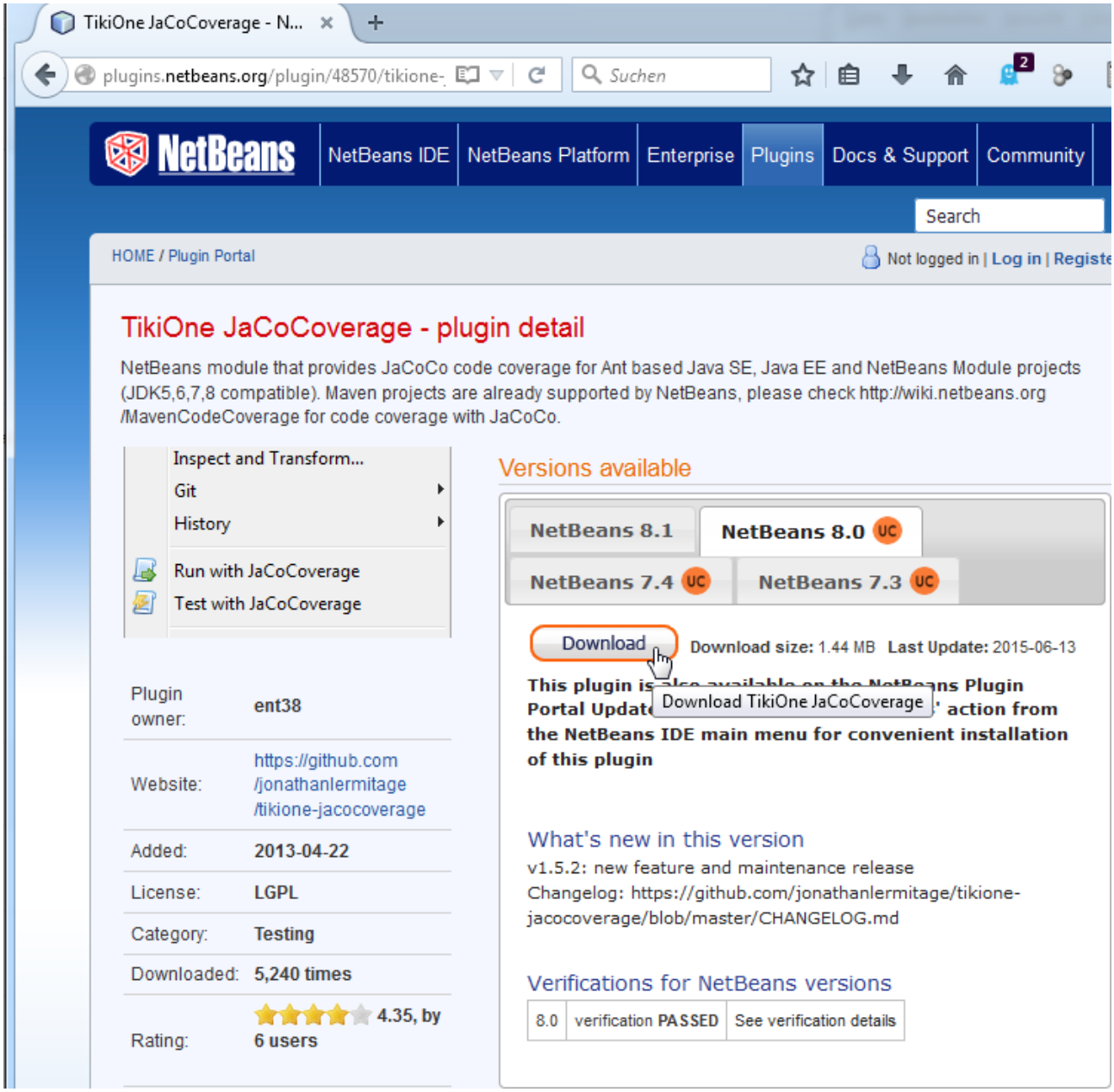
19 Überdeckungstests mit JaCoCo/TikiOne

Überdeckungstests sind ein Indikator für die Testqualität, der typischerweise nach der Erstellung der zentralen, an den Use Cases orientierten, Tests genutzt wird. Es wird berechnet, ob alle Anweisungen und alle Bedingungen durch die Tests benutzt wurden. Es ist ein guter Indikator, ob Teile der Software nicht getestet wurden. Eine hohe Überdeckung kann aber auch eine fälschliche Sicherheit sein, da selbst 100%-Abdeckung auch bei fehlerhafter Software möglich ist.

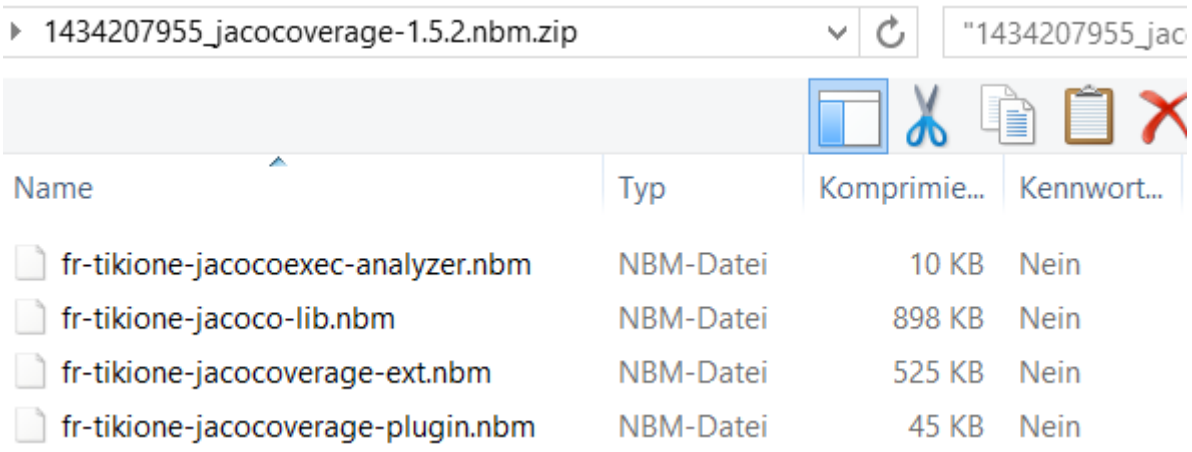
In Java gibt es zur Überdeckungsmessung u. a. die Bibliothek JaCoCo (http://eclemma.org/jacoco/), deren Nutzung mit TiKiOne in NetBeans möglich ist. TikiOne gehört zu den direkt von NetBeans unterstützen PlugIns und kann wie in „15 Installation von PlugIns - MongoDB Client“ beschrieben, installiert werden.



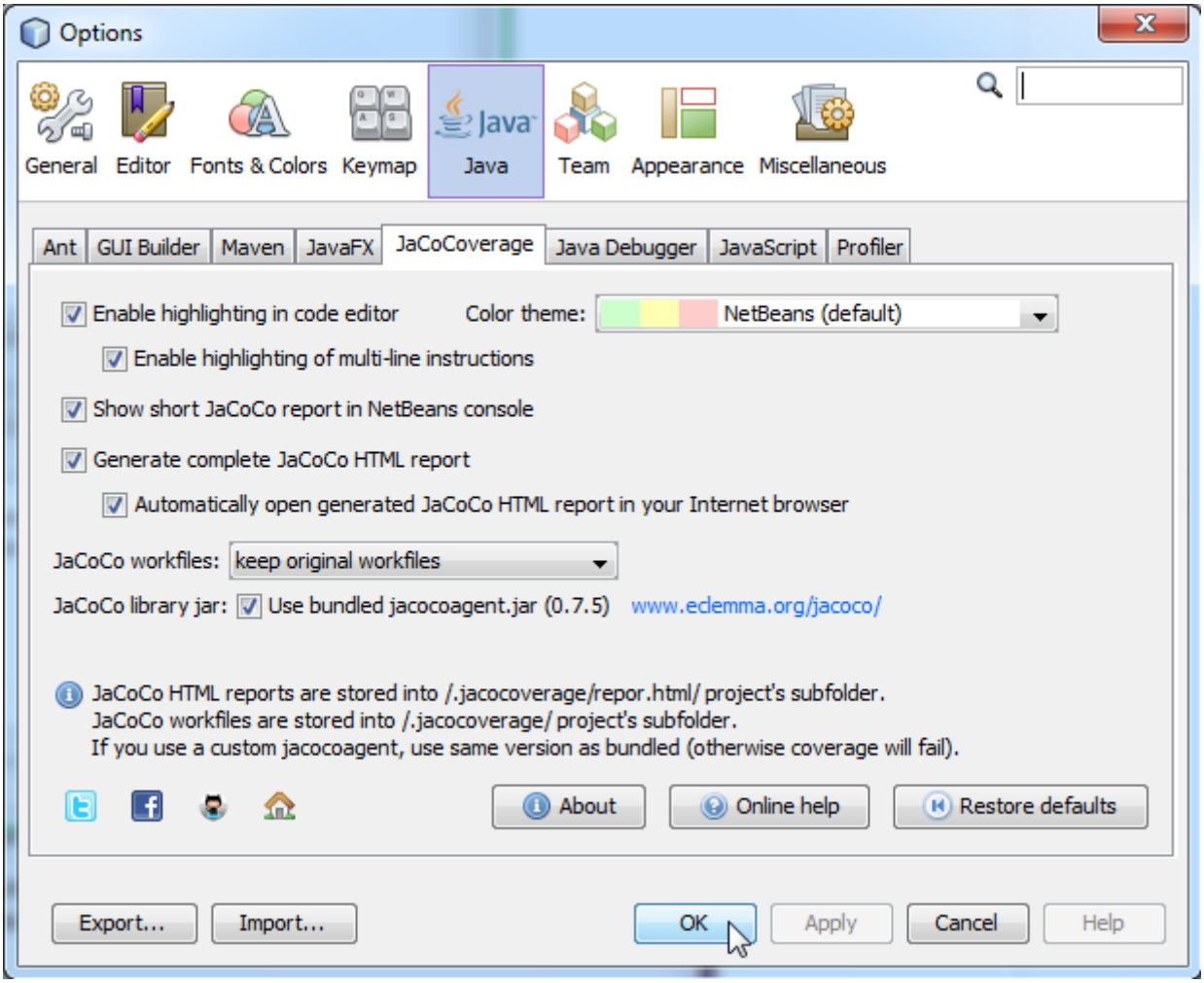
Falls die Erweiterung nicht angezeigt wird, kann sie manuell installiert werden, wie es in „15.4 Installation eines heruntergeladenen Plugins - NetBeans Command Shortcuts“ beschrieben ist. Der Download erfolgt über den Download-Button auf http://plugins.netbeans.org/plugin/48570/tikione-jacocoverage oder bei Problemen über http://sourceforge.net/projects/tikione/.



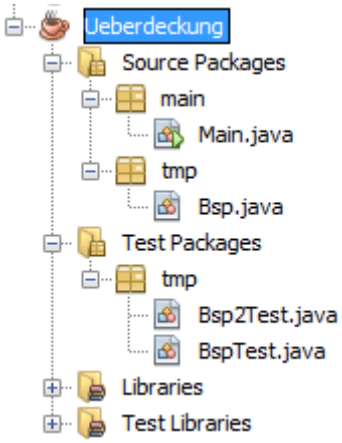
Das erhaltene Zip-File wird ausgepackt und die vier mit *.nbm endenden Files in NetBeans installiert.



Unter „Tools -> Options“, dem oberen Feld „Java“ und dem Reiter „JaCoCoverage“ kann die Erweiterung konfiguriert werden. Es besteht aber meist keine Notwendigkeit die Standardeinstellungen zu ändern.



Als Beispiel wird ein Projekt mit den folgenden Klassen betrachtet, die Tests könnten natürlich auch in einer Klasse stehen.




```
package main;

import tmp.Bsp;

public class Main {
    public static void main(String[] s){
        Bsp bsp = new Bsp();
        System.out.println("42: " + bsp.vorzeichen(42));
    }
}


package tmp;

public class Bsp {

    public int vorzeichen(int wert){
        if (wert < 0){
            return -1;
        }
        if (wert > 0){
            return 1;
        } else {
            return 0;
        }
    }

}


package tmp;
```

```
import junit.framework.Assert;
import org.junit.Before;
import org.junit.Test;

public class BspTest {

    Bsp bsp;

    @Before
    public void setUp() {
        bsp = new Bsp();
    }

    @Test
    public void testVorzeichenPositiv(){
        int erg = bsp.vorzeichen(42);
        Assert.assertTrue("42 positiv, aber :" + erg
                , erg == 1);
    }
}


package tmp;

import junit.framework.Assert;
import org.junit.Before;
import org.junit.Test;

public class Bsp2Test {

    Bsp bsp;

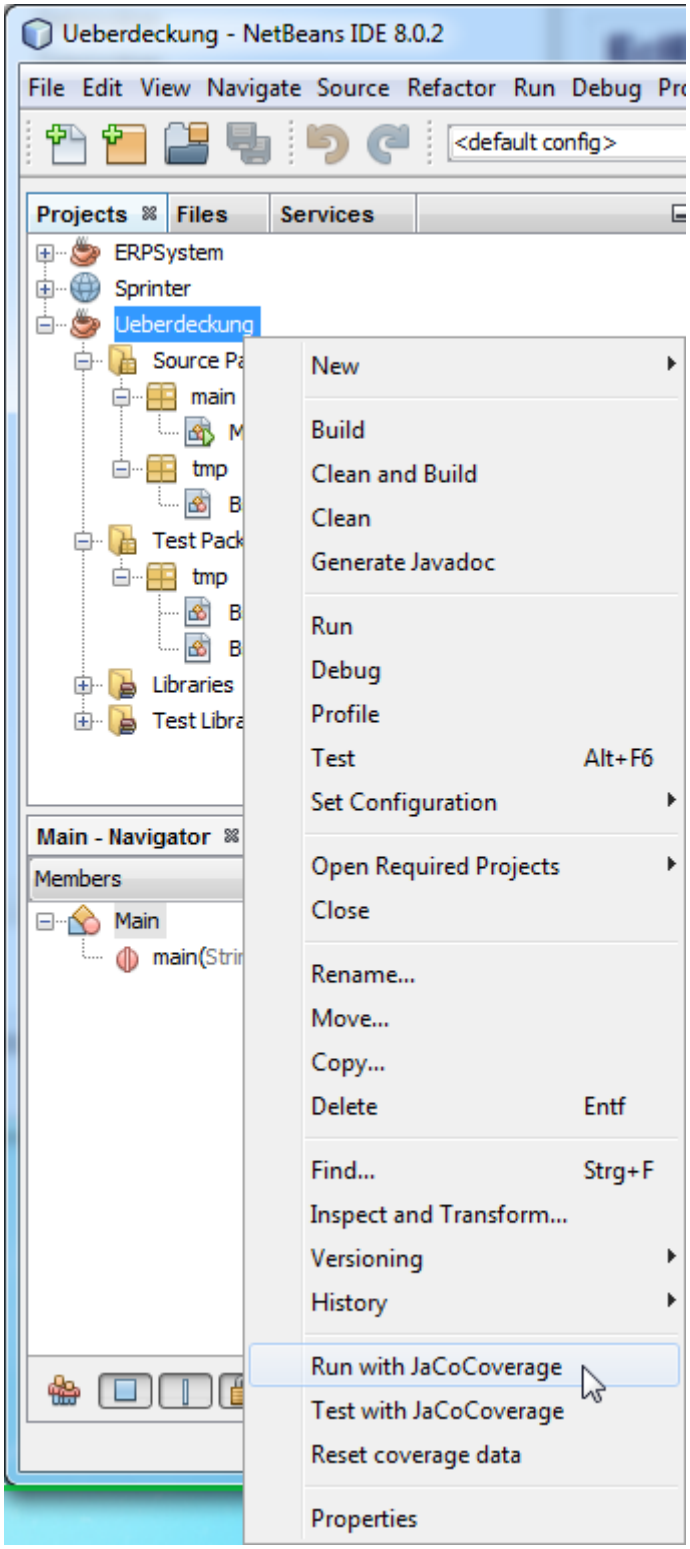
    @Before
    public void setUp() {
        bsp = new Bsp();
    }

    @Test
    public void testVorzeichenNegativ(){
        int erg = bsp.vorzeichen(-42);
        Assert.assertTrue("-42 negativ, aber :" + erg
                , erg == -1);
    }
}
```

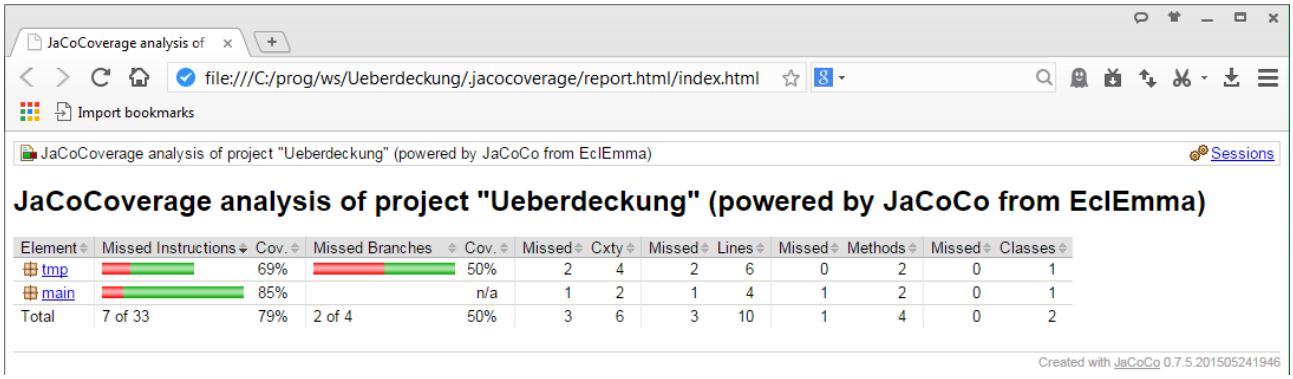
TikiOne bietet zwei Möglichkeiten zur Testausführung. Die erste besteht darin, das Programm „normal“ über seine main-Methode zu starten und dann zu protokollieren, welche Anweisungen ausgeführt wurden. Dies kann gerade bei manuellen Tests interessant sein, bei denen die

Testspezifikation in Textform vorliegt und dann manuell abgearbeitet wird. Bei der zweiten Form werden die zugehörigen Tests des Projekts ausgeführt.

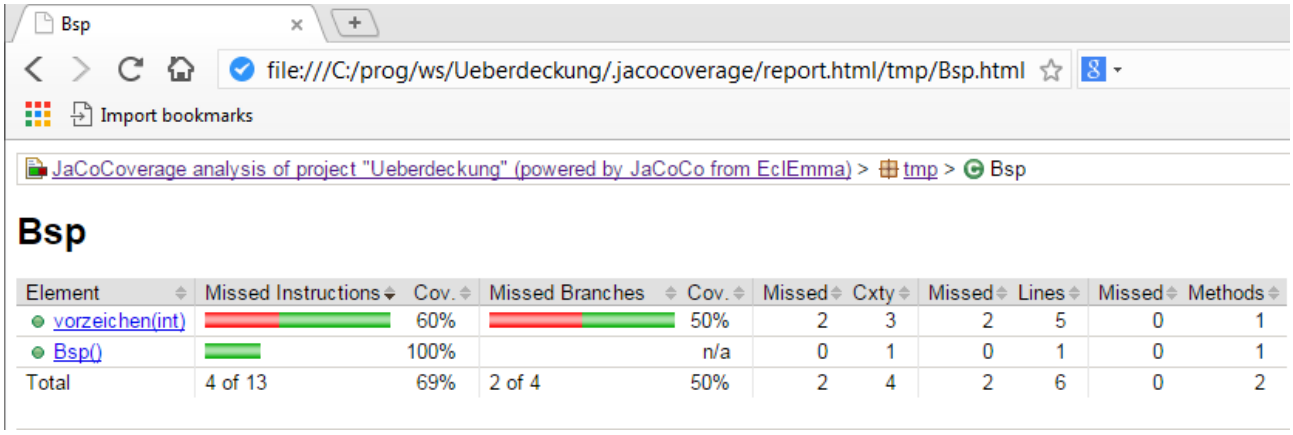
Soll das Programm mit TikiOne gestartet werden, wird ein Rechtsklick auf dem Projekt gemacht und unten „Run with JaCoCoverage“ ausgewählt. Im Beispiel wird die main-Methode dadurch gestartet.



Direkt nach dem Programmende geht der ausgewählte Web-Browser auf und zeigt folgendes Ergebnis der Überdeckung an.



Es werden alle Pakete angezeigt, dazu angegeben, wieviele Befehlszeilen und wieviele Zweige nicht ausgeführt wurden. Die Pakete können durch einen einfachen Klick aufgeklappt werden, bis die einzelnen Methoden sichtbar sind.



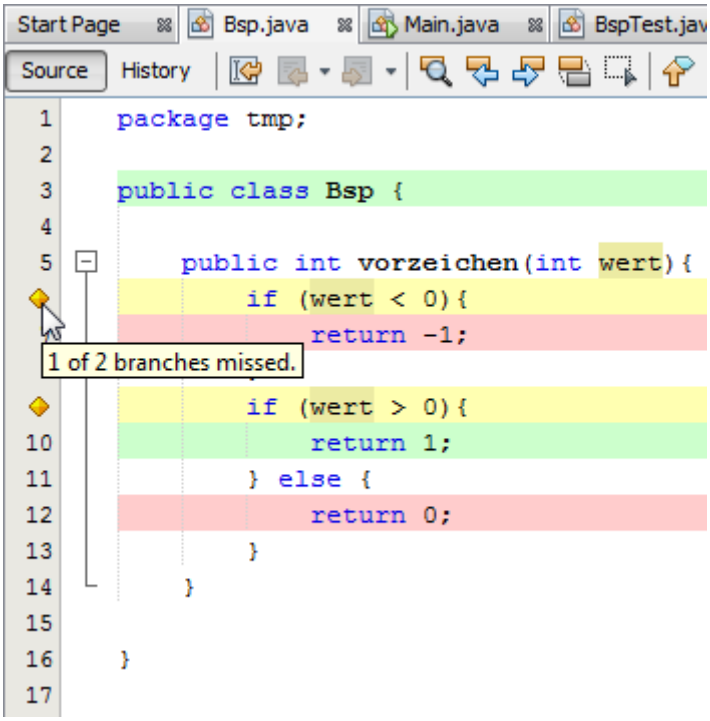
Nach einem Klick auf die Methode wird das Ergebnis der Überdeckung im Detail angezeigt. Mit der Default-Einstellung sind genutzte Zeilen grün, ungenutzte rot und Entscheidungen, bei denen nicht alle Booleschen Ausdrücke einmal true und einmal false wurden, gelb. Wird die Maus auf eine Randmarkierung bei einer Entscheidung geschoben, wird das Ergebnis noch detaillierter angezeigt.



Bei der Startmethode wird nebenbei sichtbar, dass JaCoCo Schwierigkeiten mit Klassenanfängen haben kann, so dass 100%-Ergebnisse kaum erreichbar sind. Bei größeren Programmen fällt dies aber nicht ins Gewicht.

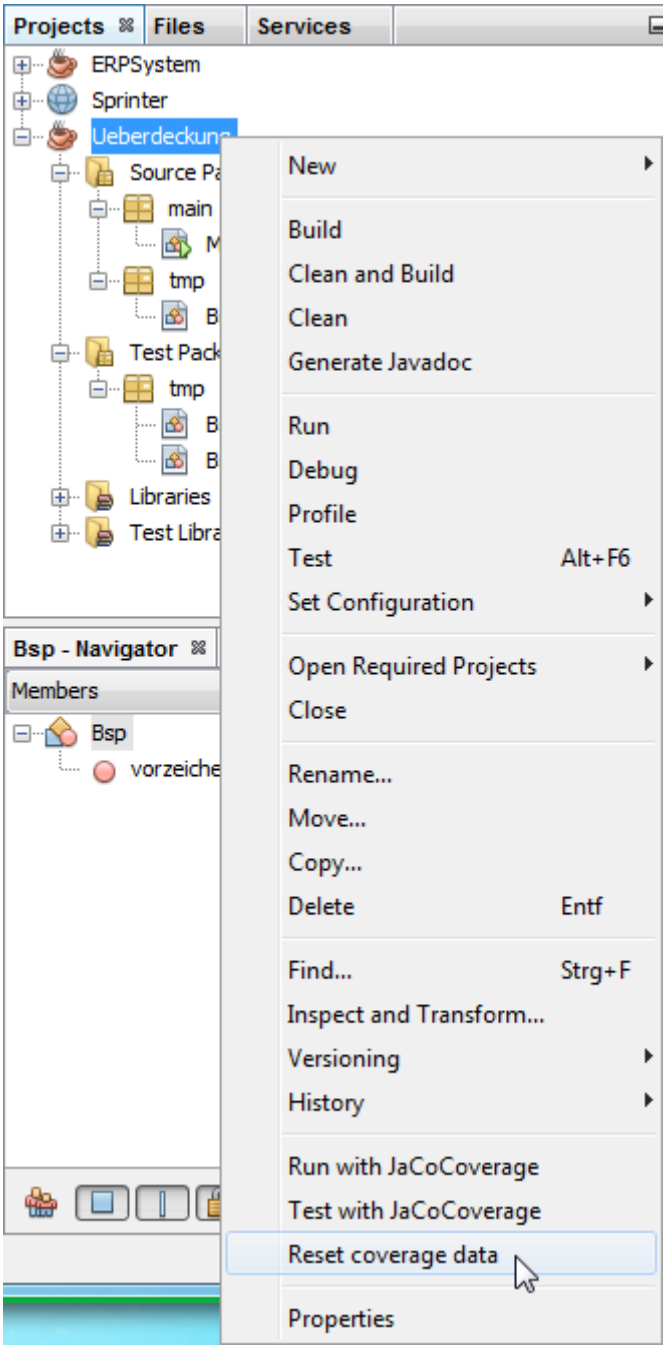


Die gleiche Einfärbung wird auch direkt in NetBeans vorgenommen.

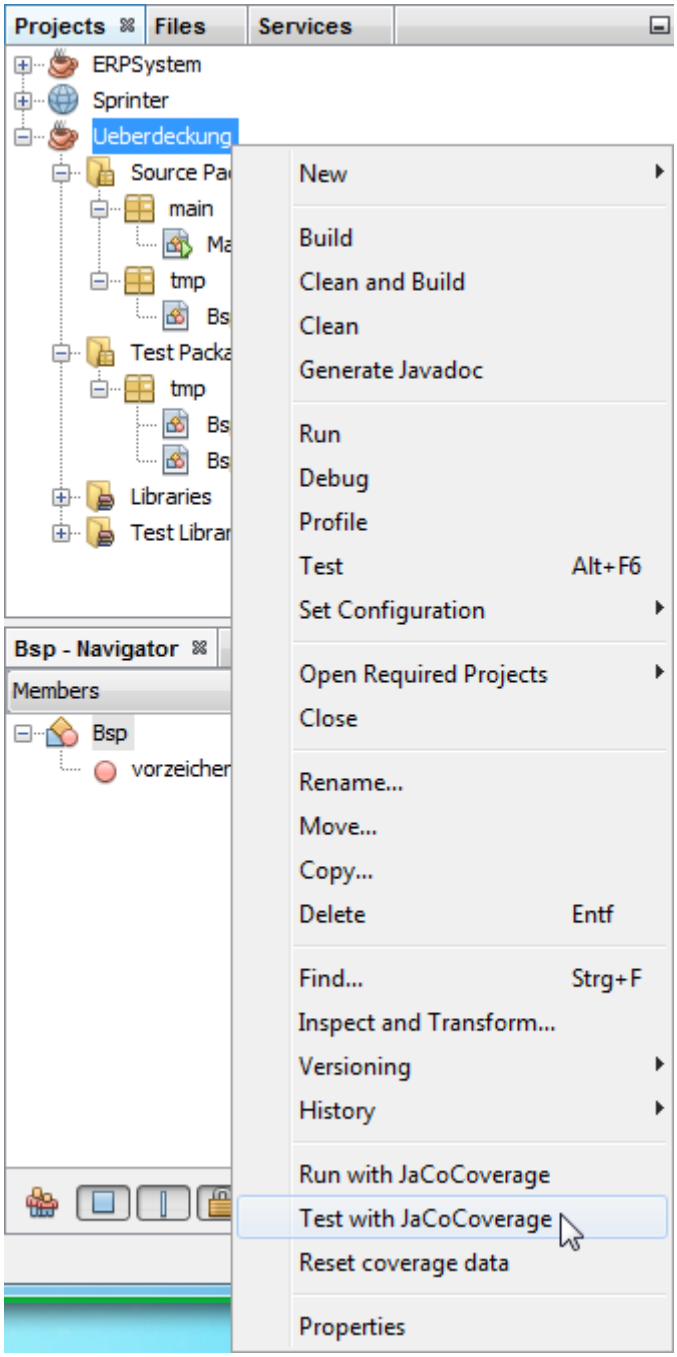



```
1   package tmp;
2
3   public class Bsp {
4
5       public int vorzeichen(int wert){
            if (wert < 0){
                return -1;
            }
            if (wert > 0){
10              return 1;
11          } else {
12              return 0;
13          }
14      }
15
16  }
17
```

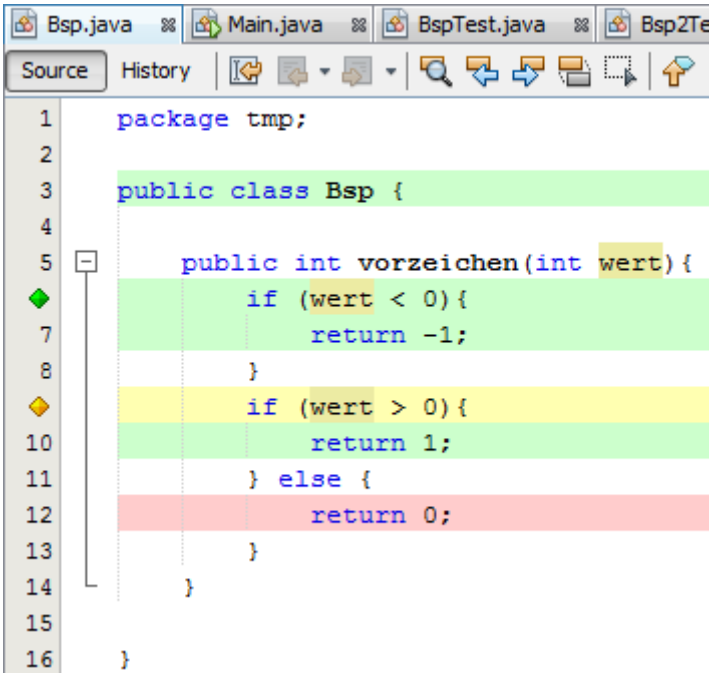
Sollen die Einfärbungen entfernt werden, wird das Projekt mit einem Rechtsklick angeklickt und „Reset Coverage Data“ gewählt.



Soll die mit den Tests erreichte Überdeckung berechnet werden, wird nach einem Rechtsklick auf dem Projekt „Test with JaCoCoverage“ ausgewählt, wodurch alle Tests ausgeführt werden, die in Klassen stehen, deren Name mit „Test“ endet. Die Auswahl einzelner Tests oder Klassen ist leider nicht möglich.



Das Ergebnis sieht wie folgt aus.

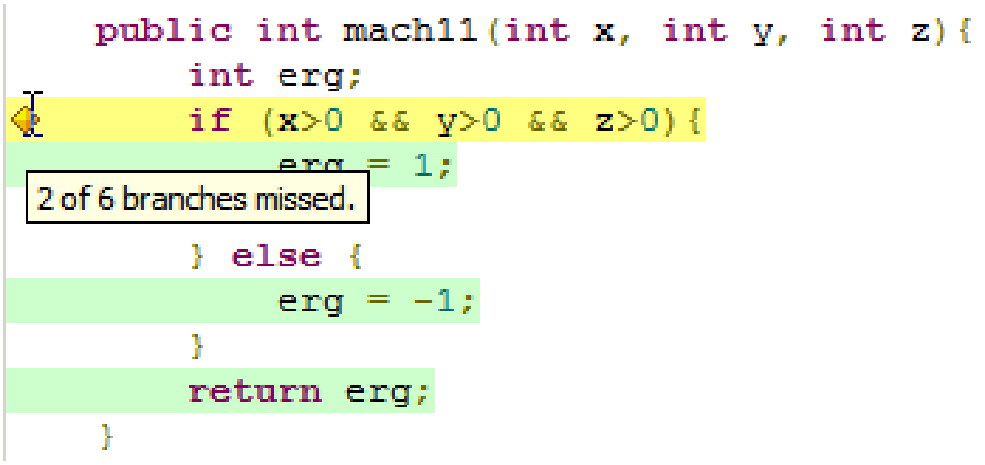


JaCoCo unterscheidet bei der Analyse von Booleschen Ausdrücken, ob eine Kurzschlussauswertung genutzt wird. Werden die nachfolgenden Testfälle

```
@Test
public void test11(){
  Assert.assertEquals(1, b.mach11(1, 1, 1));
}

@Test
public void test13(){
  Assert.assertEquals(-1, b.mach11(0, 0, 0));
}
```

auf die Methode der folgenden Abbildung angewandt, wird die dort dargestellte Überdeckung berechnet. Da es drei atomare Boolesche Ausdrücke gibt, werden sechs Branches genannt. Jeder der Ausdrücke wird im ersten Test true. Beim zweiten Test wird nur der erste Ausdruck ausgewertet, da danach bekannt ist, dass das Gesamtergebnis der Auswertung false ist. Also werden insgesamt vier der sechs Fälle betrachtet.



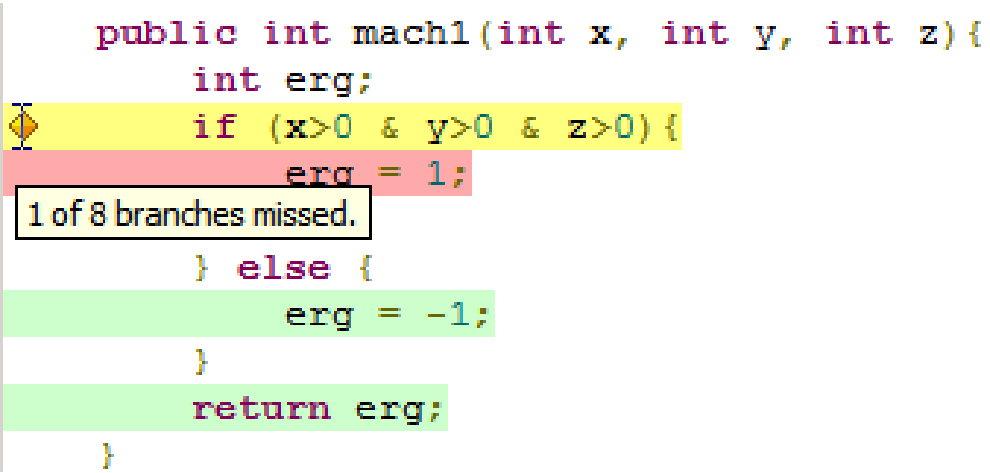
Nutzt man keine Kurzschlussauswertung, ist es theoretisch möglich, dass alle atomaren Ausdrücke einmal true und einmal false sind, der gesamte Ausdruck aber nur einen der Wahrheitswerte annimmt. Um dies zu berücksichtigen ergänzt JaCoCo im Fall der Nutzung eines der Operatoren & oder | zwei weitere Branches bei der Berechnung für das Gesamtergebnis des Ausdrucks.

Werden die die folgenden zwei Testfälle

```
@Test
public void test1(){
  Assert.assertEquals(-1, b.mach1(0, 1, 0));
}

@Test
public void test3(){
  Assert.assertEquals(-1, b.mach1(1, 0, 1));
}
```

für die in der folgenden Abbildung gezeigte Methode genutzt, wird die dort dargestellte Überdeckung berechnet. Jeder atomare Boolesche Ausdruck wird einmal false und einmal true, der Gesamtausdruck aber immer nach false ausgewertet. Dadurch fehlt insgesamt ein Branch.

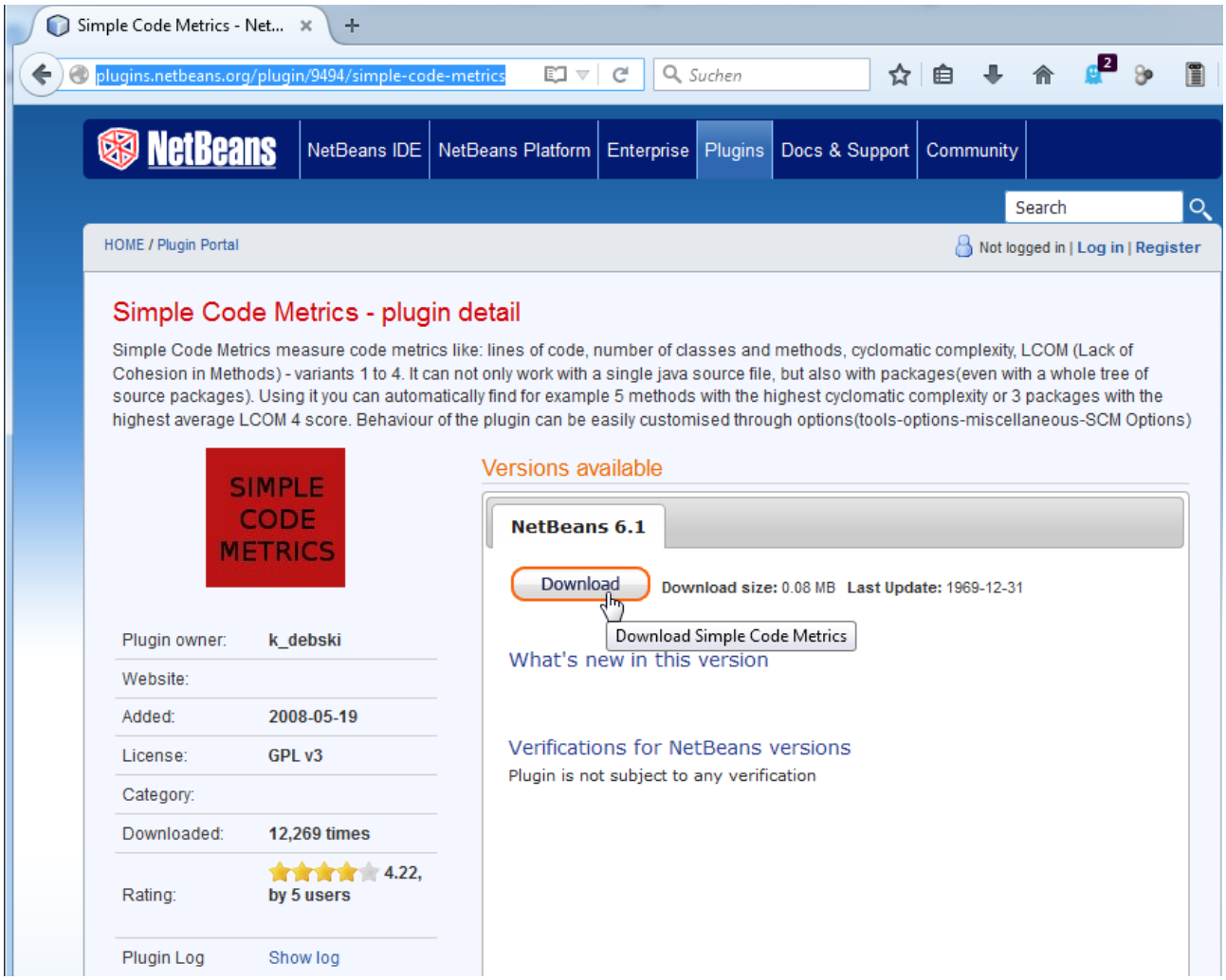


Weitere Details zu den Berechnungen in JaCoCo können der Seite http://eclemma.org/jacoco/trunk/doc/counters.html entnommen werden.

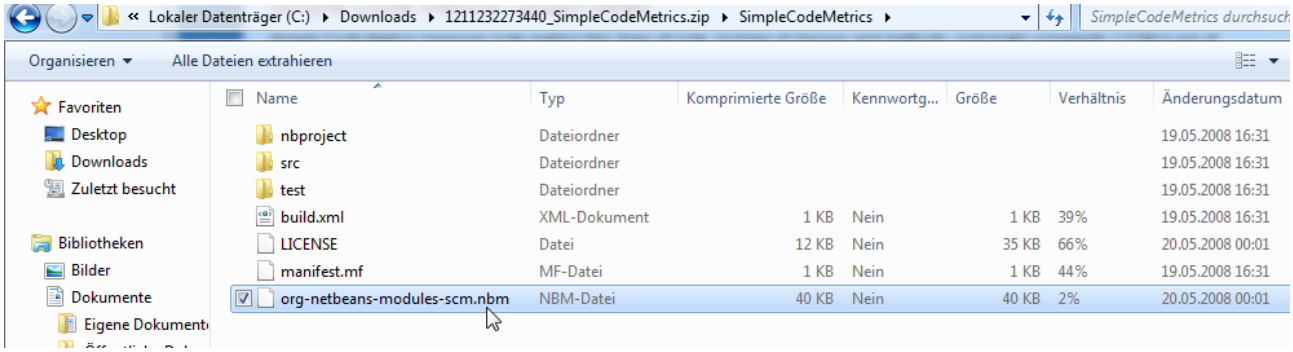
20 Metriken mit Simple Code Metrics

Metriken können Indikatoren für schlechte Software, genauer einen schlechten, also zu komplizierten, Methodenaufbau oder eine schlechte Software-Architektur liefern. Die resultierenden Ergebnisse dürfen dabei nicht überinterpretiert werden, allerdings sollten schlechte Werte begründbar sein. NetBeans bietet das Plugin Simple Code Metrics, das einige interessante Metriken in sehr einfacher Darstellung anbietet. Das Plugin wird anscheinend nicht mehr gepflegt, läuft aber zumindest auch in NetBeans 8.0.2.

Das Plugin kann von der Seite http://plugins.netbeans.org/plugin/9494/simple-code-metrics mit einem Klick auf „Download“ heruntergeladen werden.

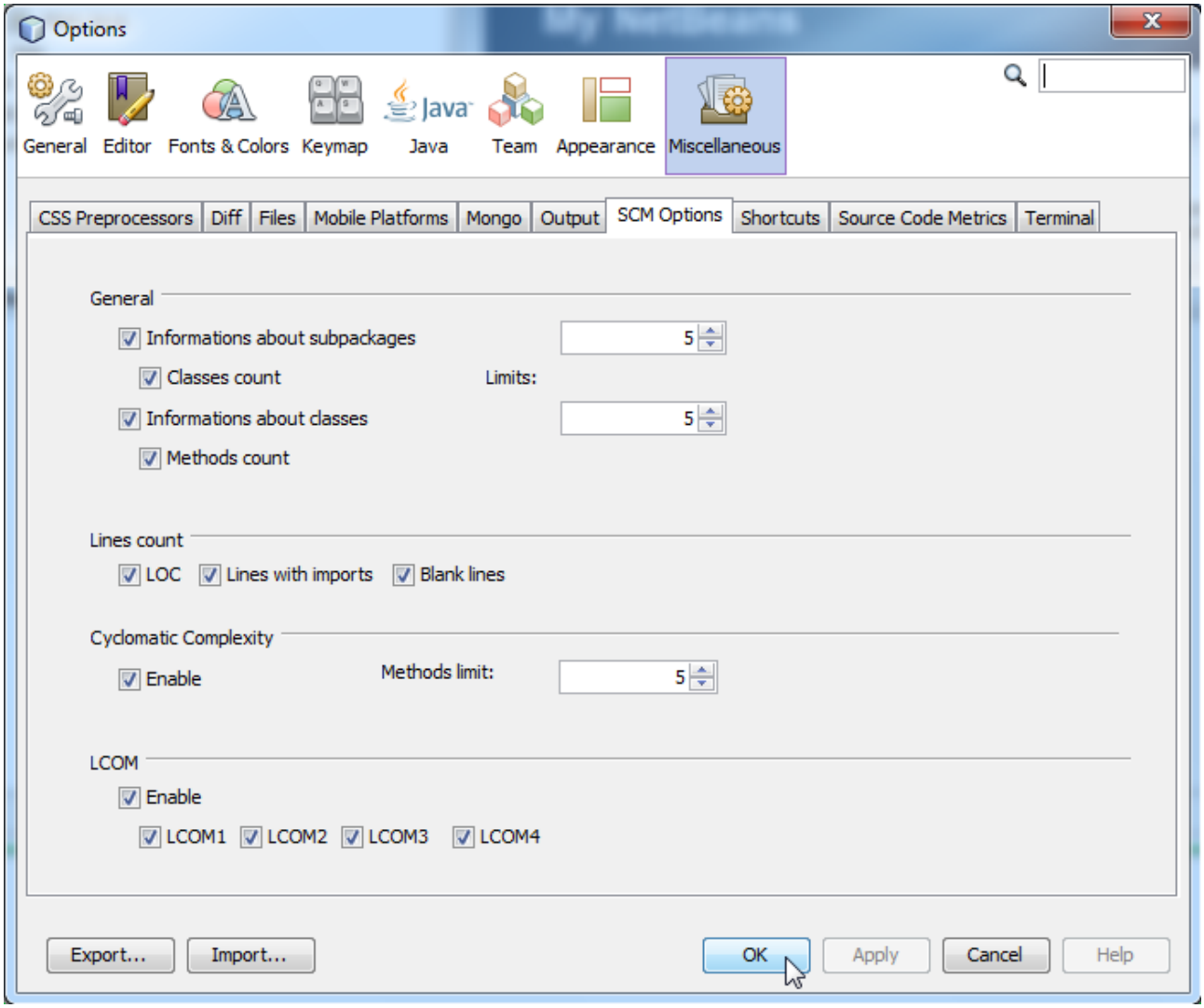


Das Plugin befindet sich dann in einem Zip-Verzeichnis, das auch den Quellcode enthält. Aus der Zip-Datei muss die Datei org-netbeans-modules-scm.nbm in ein lesbares Verzeichnis kopiert werden.

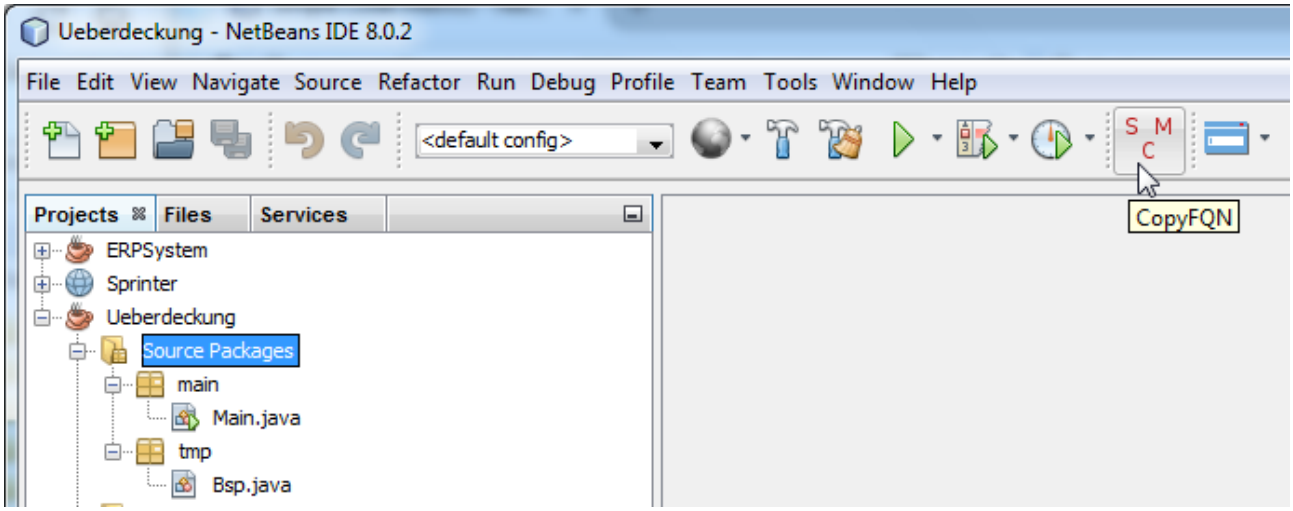


Die Installation verläuft wie bei anderen Plugins, die nicht direkt aus NetBeans heraus angeboten werden. Eine solche Installation ist in „15.4 Installation eines heruntergeladenen Plugins - NetBeans Command Shortcuts“ beschrieben.

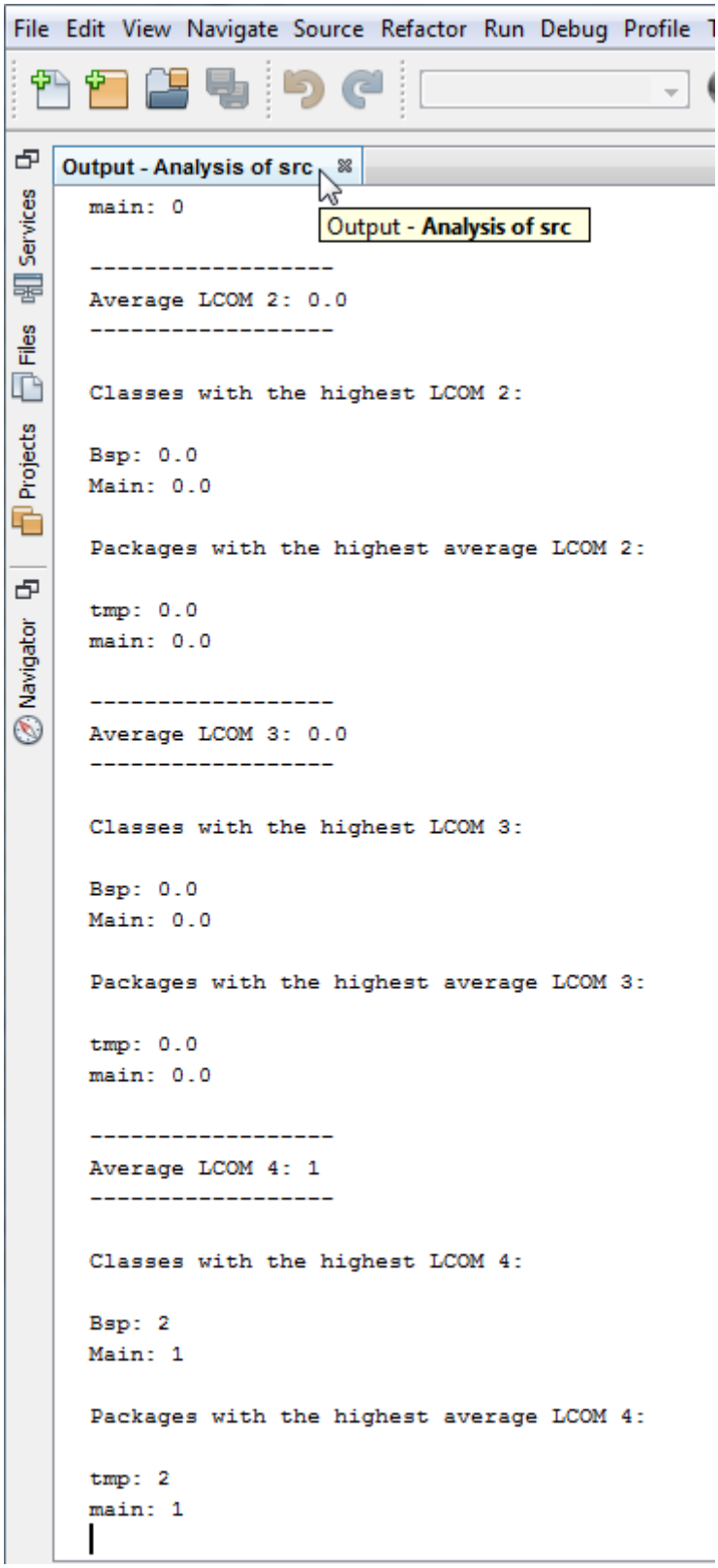
Unter „Tools -> Options -> Miscellaneous“ und dem Reiter „SCM Options“ können einige Werte eingestellt werden, generell ist das Plugin ohne Änderung nutzbar.



Die eigentliche Nutzung erfolgt über den mit „SCM“ beschrifteten Knopf, der in der oberen Menüleiste ergänzt wurde. Durch einen Klick werden die Metriken berechnet. Findet der Klick mit einer markierten Klasse statt, wird nur diese betrachtet, bei ausgewählten Paketen oder dem Ordner „Source Packages“ wird die Berechnung für alle enthaltenen Klassen durchgeführt.



Die Ergebnisse werden in einem Output-Fenster angezeigt, das für die bessere Lesbarkeit mit einem Doppelklick vergrößert werden kann.


File Edit View Navigate Source Refactor Run Debug Profile
Output - Analysis of src
Output - Analysis of src
Services
Files
Projects
Navigator
main: 0
Average LCOM 2: 0.0
Classes with the highest LCOM 2:
Bsp: 0.0
Main: 0.0
Packages with the highest average LCOM 2:
tmp: 0.0
main: 0.0
Average LCOM 3: 0.0
Classes with the highest LCOM 3:
Bsp: 0.0
Main: 0.0
Packages with the highest average LCOM 3:
tmp: 0.0
main: 0.0
Average LCOM 4: 1
Classes with the highest LCOM 4:
Bsp: 2
Main: 1
Packages with the highest average LCOM 4:
tmp: 2
main: 1

Die Metrik LCOM 3 wird auch „LCOM*“ genannt. Einige Informationen zu den Metriken können z. B. http://www.aivosto.com/project/help/pm-oo-cohesion.html entnommen werden. Das Plugin berechnet bei LCOM1 dabei statt |P|-|Q| alternativ |Q|-|P|.

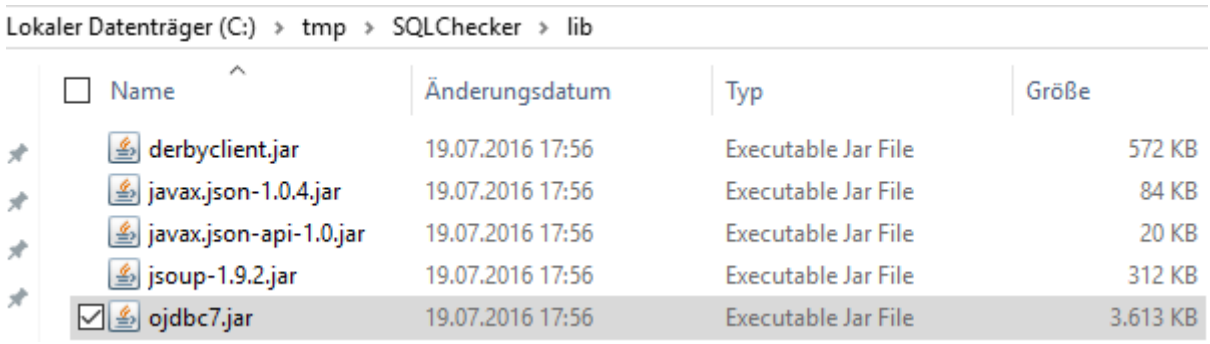
21 SQLChecker

Der SQLChecker ist ein kleines Programm, mit dem Lösungen zu SQL-Aufgaben lokal überprüft werden können. Dazu werden die Ergebnisse des eigenen Lösungsversuchs mit dem erwarteten Ergebnis verglichen. Es kann damit nicht geprüft werden, ob die Lösung sinnvoll oder effizient ist, nur dass die erreichen Ergebnisse stimmen.

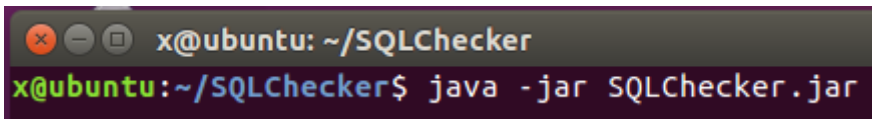
Zur Nutzung des Werkzeugs muss eine Datenbank zugreifbar sein, auf der die Anfragen laufen. Sollte der Inhalt der Datenbank nicht mit dem bei der Aufgabenerstellung genutzten Inhalt übereinstimmen, kann der SQLChecker nicht sinnvoll genutzt werden. Es ist sinnvoll, SQL-Anfragen zunächst in einem anderen Werkzeug auszuprobieren, da diese meist Syntax-Highlighting und etwas genauere Fehlermeldungen bieten. Die Lösung ist dann zur Überprüfung in dieses Werkzeug zu kopieren.

Der SQLChecker benötigt eine Java-Version ab der Version 1.8.60. Die Datei SQLChecker.zip kann an einem fast beliebigen Ort ausgepackt werden, es muss nur sichergestellt sein, dass der Nutzer in dem entstehenden Verzeichnis Lese- und Schreibrechte hat, da ein lokaler Ordner data beim ersten Aufruf zur Verwaltung der Daten angelegt wird.

Der SQLChecker enthält bereits den JDBC-Treiber für Apache Derby. Soll Oracle genutzt werden, muss die zugehörige Treiberdatei in das Unterverzeichnis lib kopiert werden.

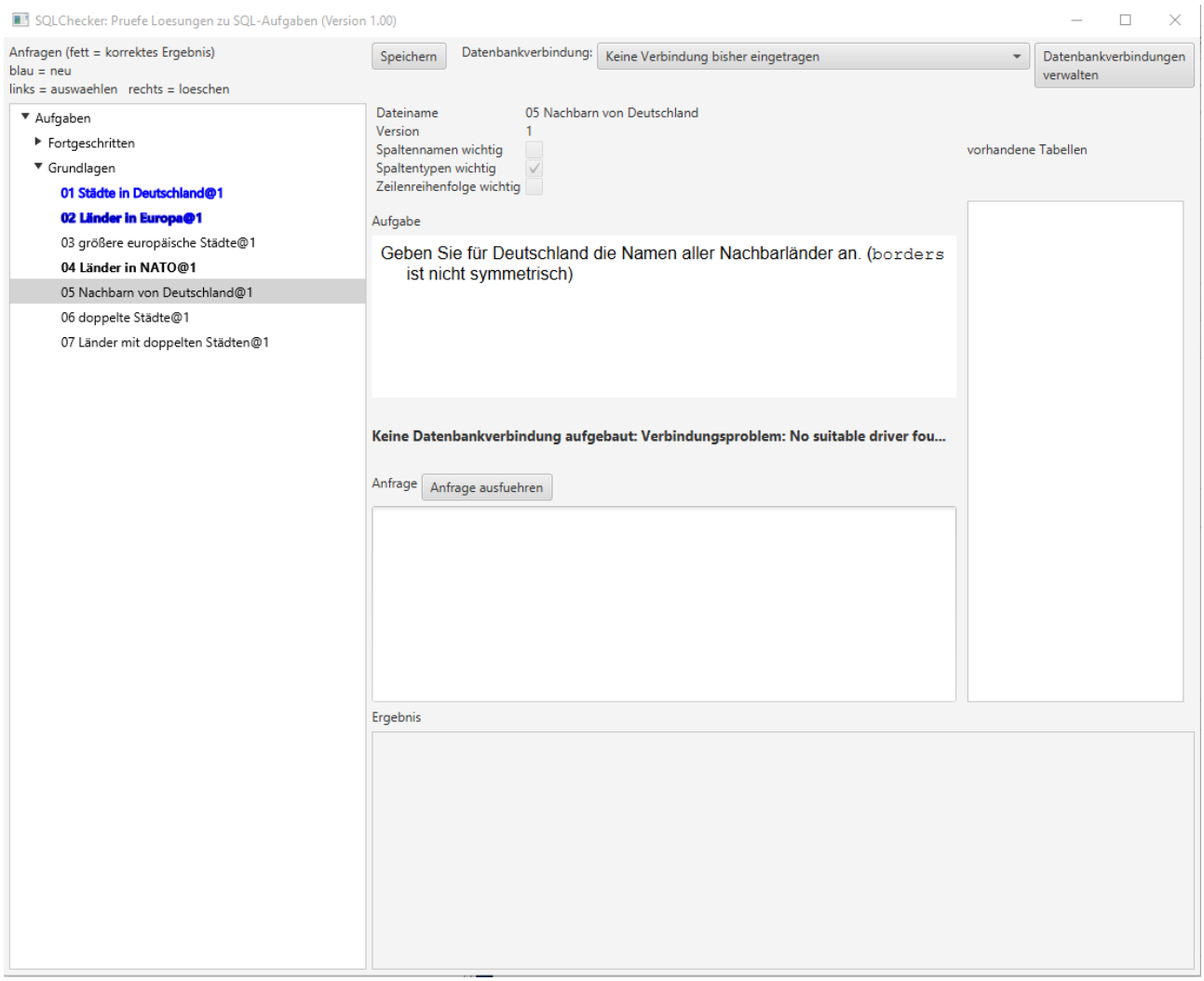


Nach dem Auspacken der Zip-Datei ist unter Windows SQLChecker.jar direkt durch einen Doppelklick startbar. Unter anderen Systemen muss in einer Konsole in das Verzeichnis manövriert und `java -jar SQLChecker.jar` aufgerufen werden. Dies gilt auch unter Ubuntu, damit das Verzeichnis data im passenden Ordner angelegt wird. Generell bietet der Aufruf unter der Konsole den Vorteil, dass der virtuellen Java-Maschine mehr Speicher über Parameter zugeordnet werden kann.

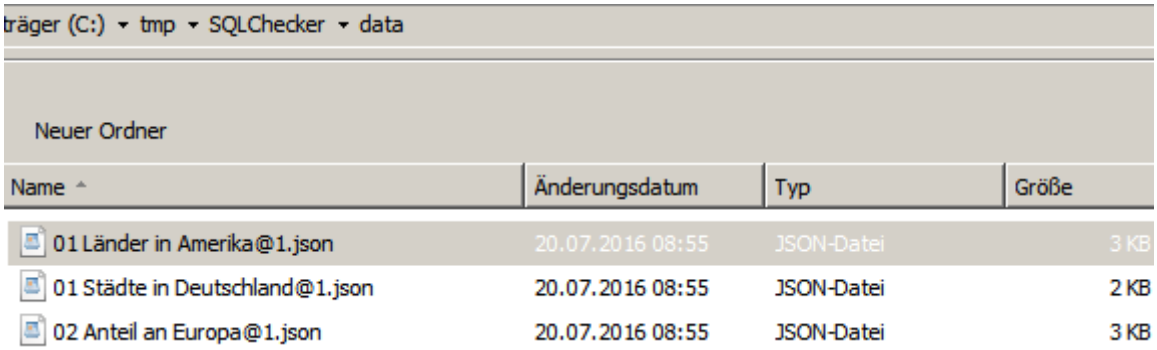


Das Werkzeug lädt zunächst alle lokal vorhandenen Aufgaben und sucht dann im Internet nach neueren Aufgaben, die ebenfalls geladen werden. Die Aufgaben werden in einem Baum auf der linken Seite angezeigt. Schwarze Dateinamen stehen für lokal geladene Aufgaben, blaue für Dateien, die bei diesem Aufruf aus dem Internet geladen wurden und jetzt lokal zur Verfügung stehen. Ist der Text fetter als normal handelt es sich um eine gelöste Aufgabe. Mit einem Links-Klick wird eine Aufgabe ausgewählt, mit einem Rechts-Klick wird sie endgültig gelöscht.

Die Namen der Aufgaben enthalten am Ende immer ein @-Symbol und eine Versionsnummer. Sollte es zwei Anfragen im Baum mit gleichen Namen geben, ist davon auszugehen, dass es mehrere Versionen der Aufgabe gibt, von denen üblicherweise die neueste bearbeitet wird.



Alle vorhandenen Aufgaben stehen im Unterverzeichnis „data“ und haben die Endung „.json“. Anders heruntergeladene Aufgaben sind zur Nutzung einfach in dieses Verzeichnis zu kopieren.



Bei der ersten Nutzung muss zunächst eine Datenbankverbindung eingerichtet werden, dazu wird der Knopf „Datenbankverbindungen verwalten“ geklickt. Um eine Datenbankverbindung auszuwählen, wird diese in der unteren Liste mit einem Linksklick angeklickt. Kann die Verbindung hergestellt werden, wird das Auswahlfenster automatisch geschlossen und die gewählte Verbindung als Datenbankverbindung in der oberen Kopfzeile angezeigt. Kann keine Verbindung hergestellt werden, wird die Auswahl in das obere Eingabefeld übernommen und ist editierbar. Der JDBC-Connection-String kann auch direkt in der oberen Zeile eingegeben werden. Durch Drücken der „Return“-Taste oder Nutzung des „Prüfe Verbindungsdaten und hinzufügen“-Knopfes wird versucht, die Verbindung aufzubauen. Ist dies erfolgreich, erfolgt ein Eintrag in der unteren Liste. Um diese Verbindung dann zu nutzen, muss diese wieder mit einem Links-Klick ausgewählt werden. Mit einem Rechts-Klick wird die geklickte Verbindung gelöscht.

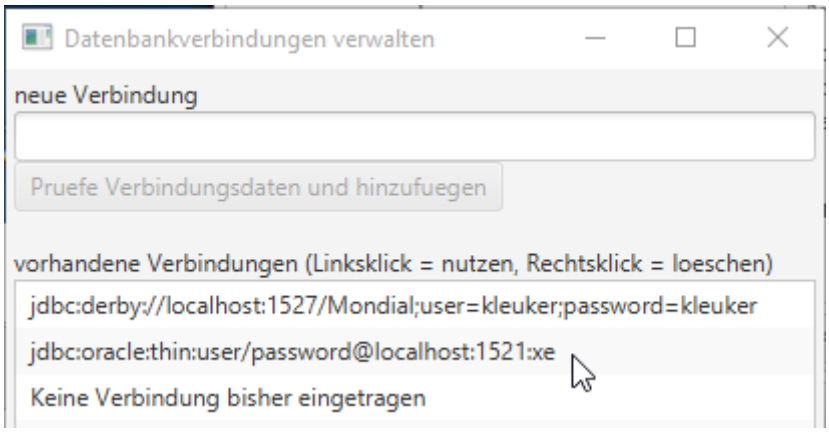
Leider sind die Ansätze unterschiedlich, wie der Nutzername und das Passwort im Connection-String ergänzt werden. Zwei Beispiele, die ohne eckige Klammern zu nutzen sind, sehen wie folgt für lokale Installationen von Derby und Oracle XE aus.

Apache Derby: `jdbc:derby://localhost:1527/Mondial;user=[name];password=[passwort]`

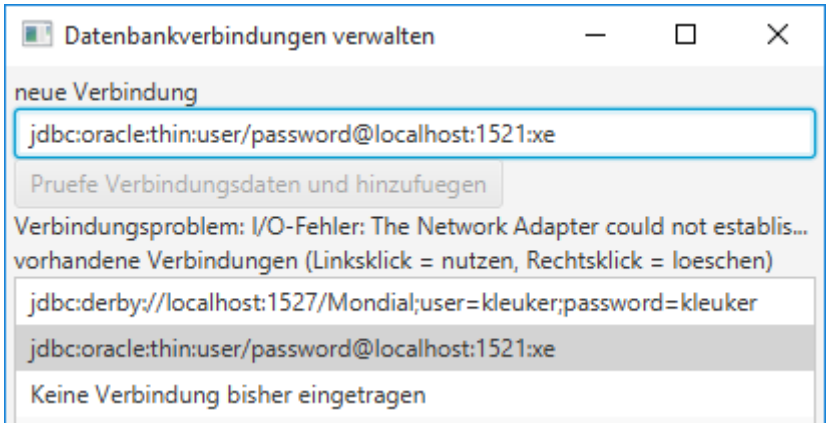
Oracle: `jdbc:oracle:thin:[name]/[passwort]@localhost:1521:xe`

Es ist zu beachten, dass Nutzername und Passwort bei diesem Werkzeug immer sichtbar sind. Die folgenden Abbildungen zeigen den erfolglosen und dann den erfolgreichen Versuch, eine Datenbankverbindung zu ergänzen.

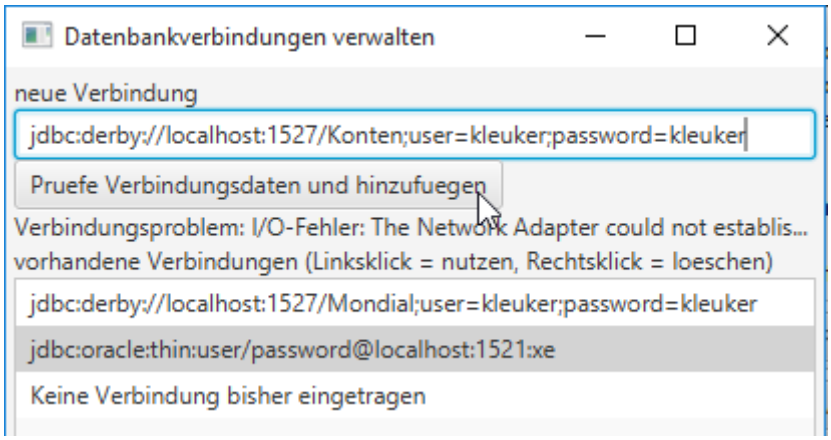
Linksklick auf einen String.



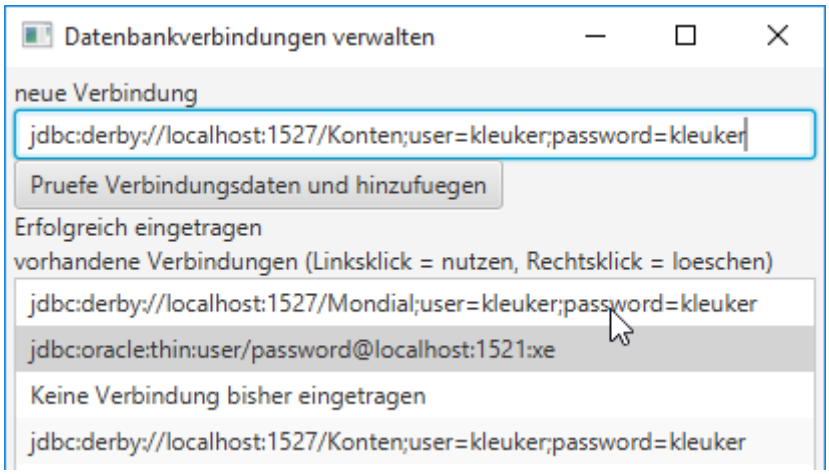
Es gibt einen kurzen Fehlerhinweis, weiterhin kann die Eingabe oben editiert werden. Bei Oracle dauert eine Verbindungsprüfung und der spätere Neuaufbau etwas länger.



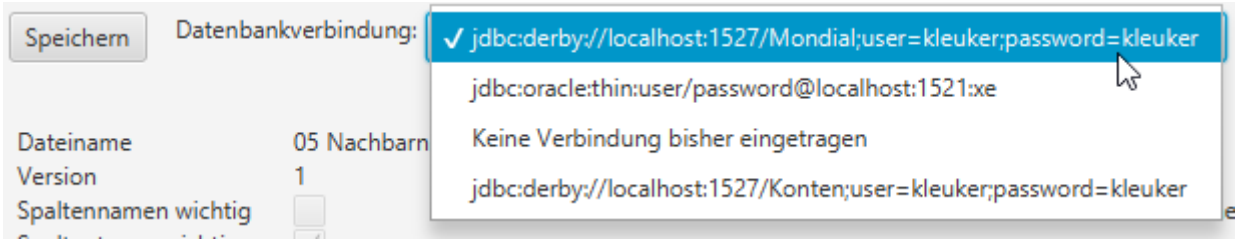
Es wird ein neuer String eingegeben und „Return“ gedrückt.



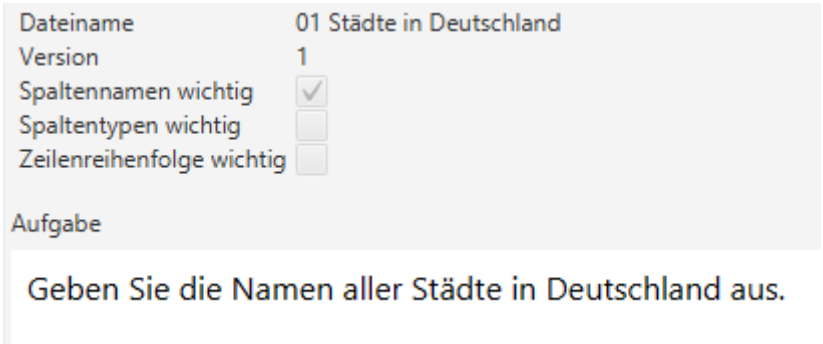
Der String wurde unten ergänzt. Mit einem Linksklick wird eine Verbindung ausgewählt.



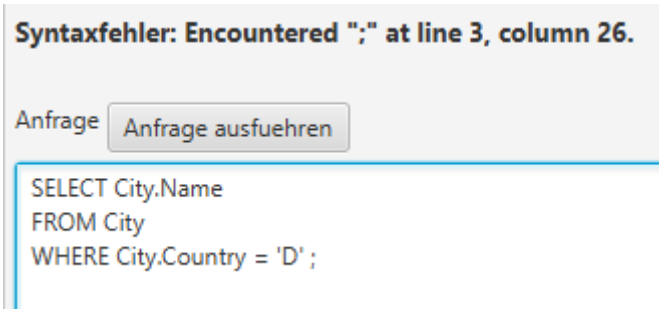
Die ausgewählte Verbindung wird oben angezeigt und ist auch in der Drop-Down-Box änderbar. Auf der rechten Seite werden zur Information die vorhandenen Tabellen angezeigt, die Anzeige ist nicht weiter nutzbar.



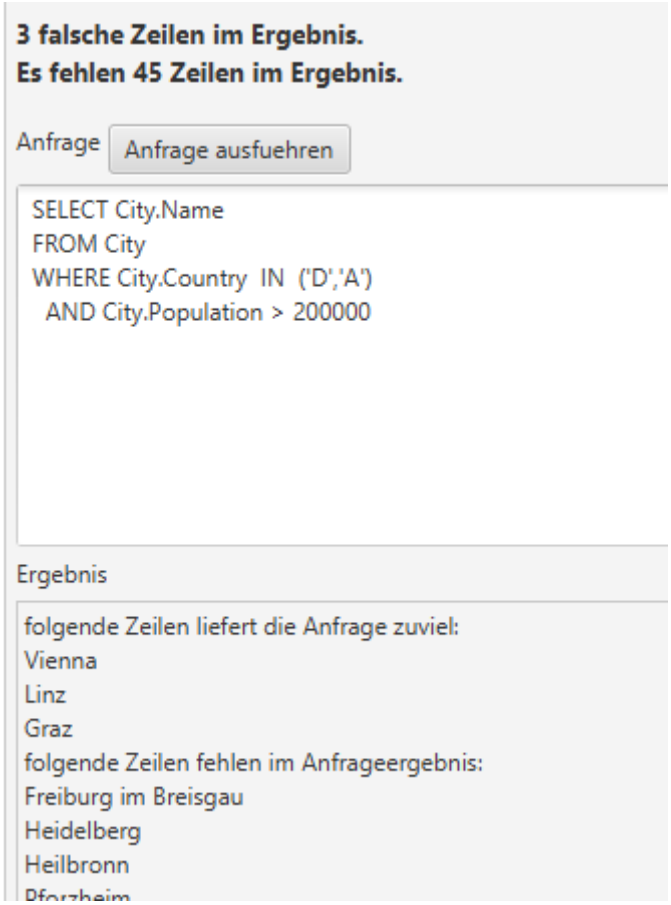
Im Aufgabe-Feld steht die zu bearbeitende Aufgabe, dabei können die oberen drei CheckBoxen interessant sein. Ein Haken bei „Spaltennamen wichtig“ bedeutet, dass der genaue Spaltenname zur Lösung passen muss. Steht dazu nichts in der Aufgabenstellung, sind typischerweise die Spaltennamen der genutzten Tabellen gemeint, so dass hier keine besondere Beachtung notwendig ist. Ein Haken bei „Spaltentypen wichtig“ bedeutet, dass konkrete Typen für die Ergebnisspalten gefordert sind, so dass z. B. nicht eine Spalte mit korrektem Namen, aber statt INTEGER dem Typ VARCHAR in der Lösung genutzt wird. Für die konkrete Aufgabenstellung ist aber ein solcher Haken meist vernachlässigbar, da er eine Einschränkung auf ein konkretes Datenbanksystem sein kann, da z. B. Oracle mehr Zahlentypen mit etwas anderen Wertebereichen unter dem Namen Number anbietet, als es in Derby der Fall ist. Ein Haken bei „Zeilenreihenfolge wichtig“ bedeutet, dass die Reihenfolge in der Ergebnisaushabe relevant ist. Da SQL mengenorientiert arbeitet, muss die gleiche Anfrage nicht zur gleichen Reihenfolge der Ergebniszeilen führen.



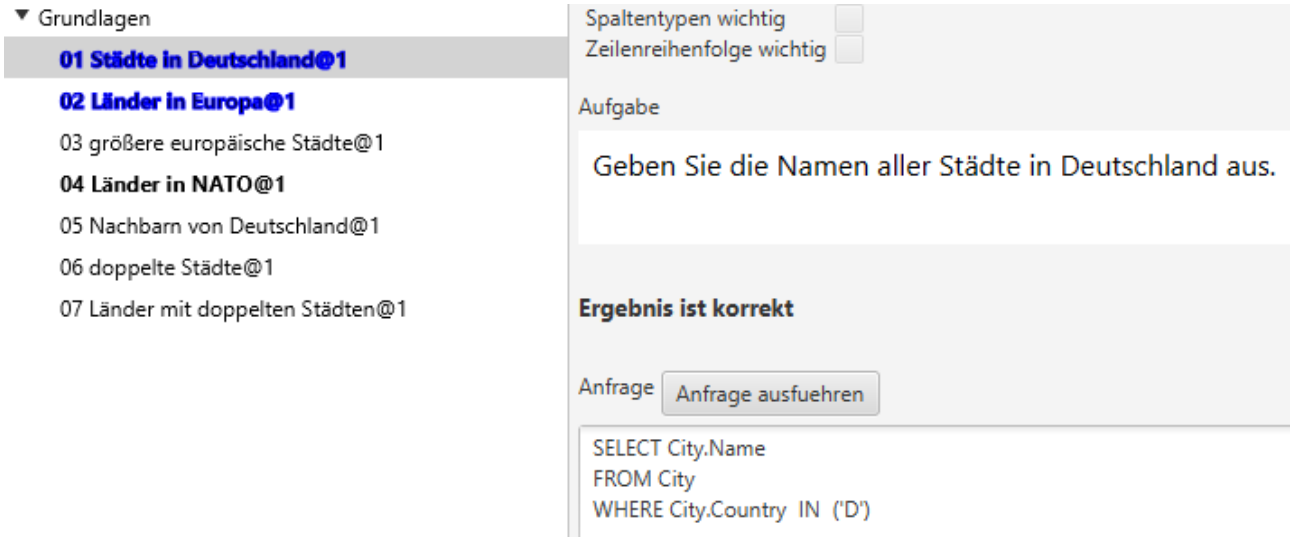
Der eigentliche Lösungsversuch wird im Feld „Anfrage“ eingegeben und mit dem Knopf „Anfrage ausfuehren“ ausgeführt. In dem Informationsfeld darüber werden etwaige Hinweise ausgegeben. Im folgenden Beispiel liegt ein Syntaxfehler vor.



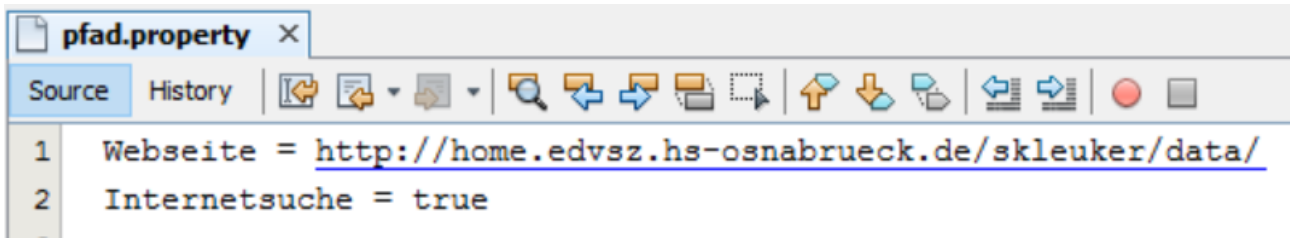
Sollte die Anfrage Zeilen enthalten, die nicht zum Ergebnis gehören oder es Zeilen im gewünschten Ergebnis geben, die das Ergebnis der Anfrage nicht liefert, werden diese im unteren Ergebnisfenster angezeigt.



Liefert die Anfrage das korrekte Ergebnis, wird dies im Informationsfeld ausgegeben und die Anfrage wird im Baum fetter dargestellt. Weiterhin wird eine korrekte Lösung automatisch gespeichert. Alle anderen Lösungsversuche müssen über den Knopf „Speichern“ explizit abgespeichert werden, ansonsten gehen alle eingetragenen Informationen beim Aufruf einer neuen Aufgabe endgültig verloren.



Das Standardverhalten vom SQLChecker kann durch eine Datei „pfad.property“, die im Verzeichnis „data“ liegen muss, etwas modifiziert werden. In der Eigenschaft „Webseite“ steht die Webseite, genauer das Verzeichnis, in dem nach Aufgaben gesucht wird. Mit dem Booleschen Wert der Eigenschaft „Internetsuche“ ist festlegbar, ob überhaupt eine Suche im Internet nach Aufgaben ausgeführt werden soll.

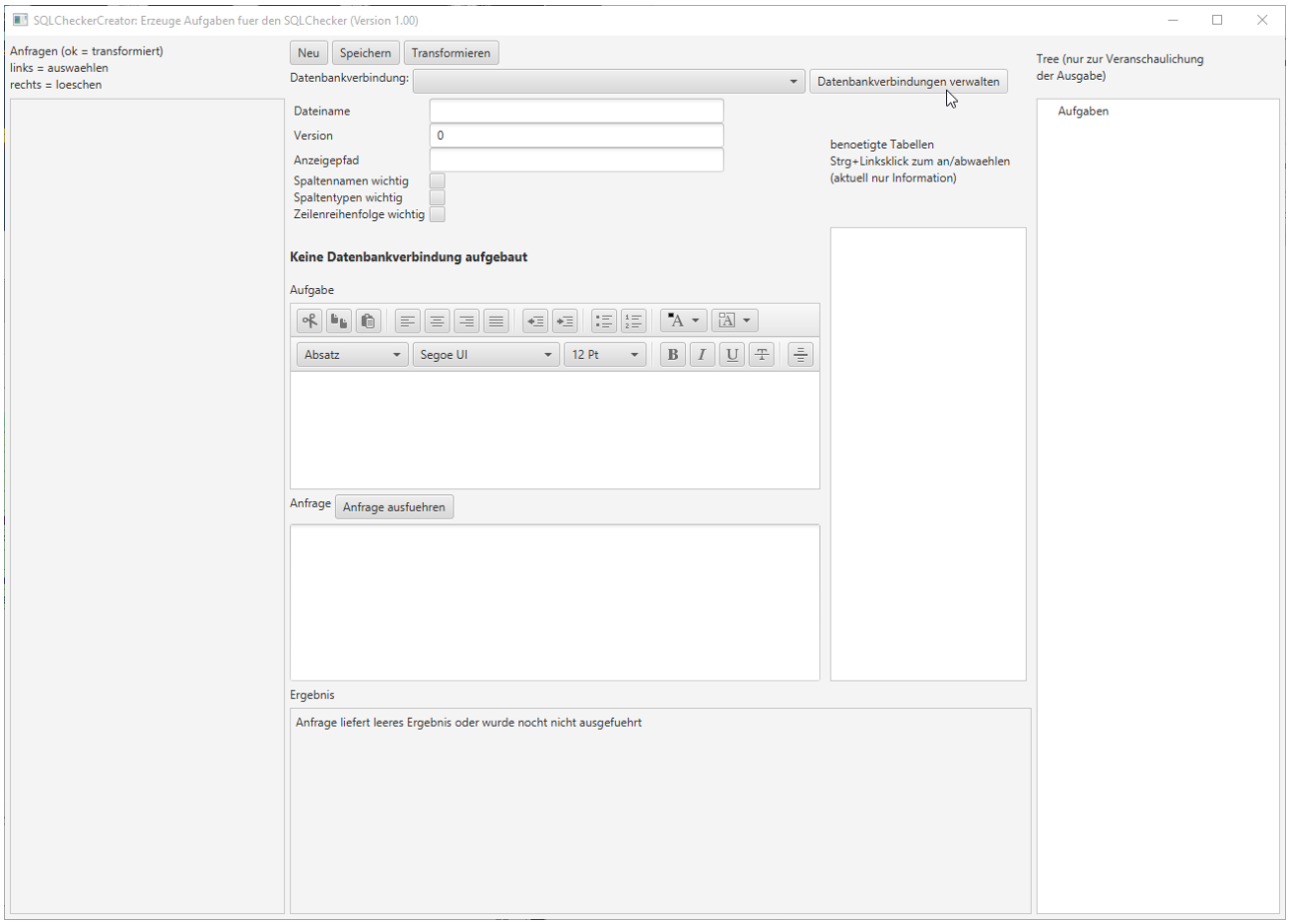


22 SQLCheckerCreator

Der SQLCheckerCreator dient zur Erstellung von Aufgaben für den SQLChecker. Zum Verständnis dieses Kapitels muss zuerst das Kapitel über den SQLChecker gelesen werden.

Die Installationsbedingungen für den SQLCheckerCreator sind identisch mit denen des SQLCheckers. Wichtig ist dabei nur, dass sich nicht beide Werkzeuge im gleichen Verzeichnis befinden, da es sonst durch ein gemeinsam genutztes Verzeichnis „data“ zu Problemen kommen kann.

Die Verbindung zu einer Datenbank wird genau wie beim SQLChecker hergestellt.

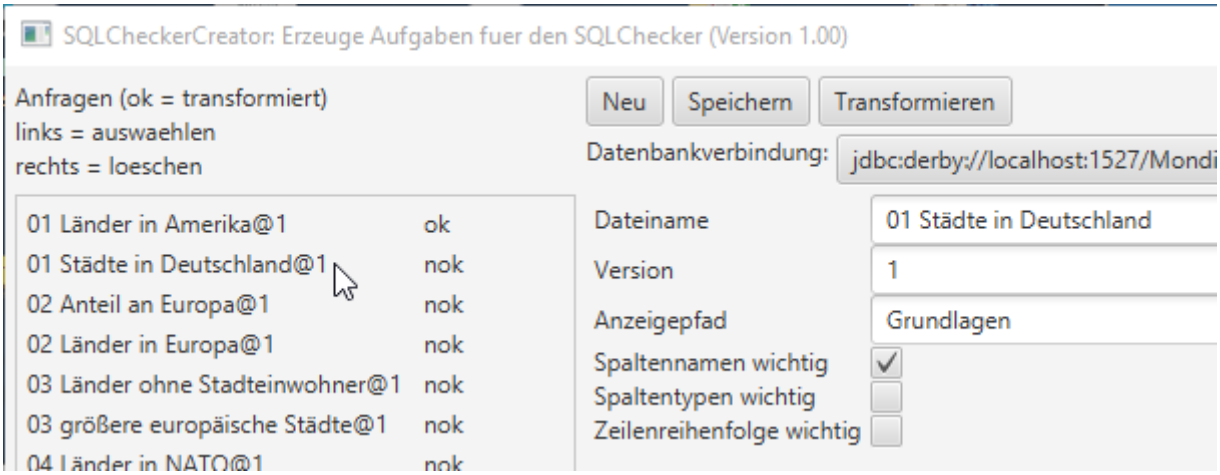


Die Rahmenbedingungen für die Aufgabenstellung werden oben in der Mitte definiert. Unter „Dateiname“ muss ein für dieses Werkzeug eindeutiger Name angegeben werden, sollte der Namen schon vorhanden sein, wird eine ältere Version ohne Hinweis gelöscht. Die „Version“ kann beliebig vergeben werden und wird mit in den Dateinamen aufgenommen, so dass es verschiedene Versionen der gleichen Aufgabe geben kann. Der Anzeigepfad ist „virtuell“, er wird zur Anzeige in der vom SQLChecker genutzten Baumstruktur verwendet. Der Eintrag darf nicht leer sein, alle Ebenen werden durch einen „/“ voneinander getrennt. Die CheckBoxen sind beim SQLChecker erklärt, ein Haken bei „Spaltentypen wichtig“ kann dazu führen, dass die Aufgabe nur mit einem Datenbanksystem

bearbeitbar ist. Vorhandene Anfragen werden auf der linken Seite angezeigt, mit einem Links-Klick ausgewählt und mit einem Rechts-Klick gelöscht.

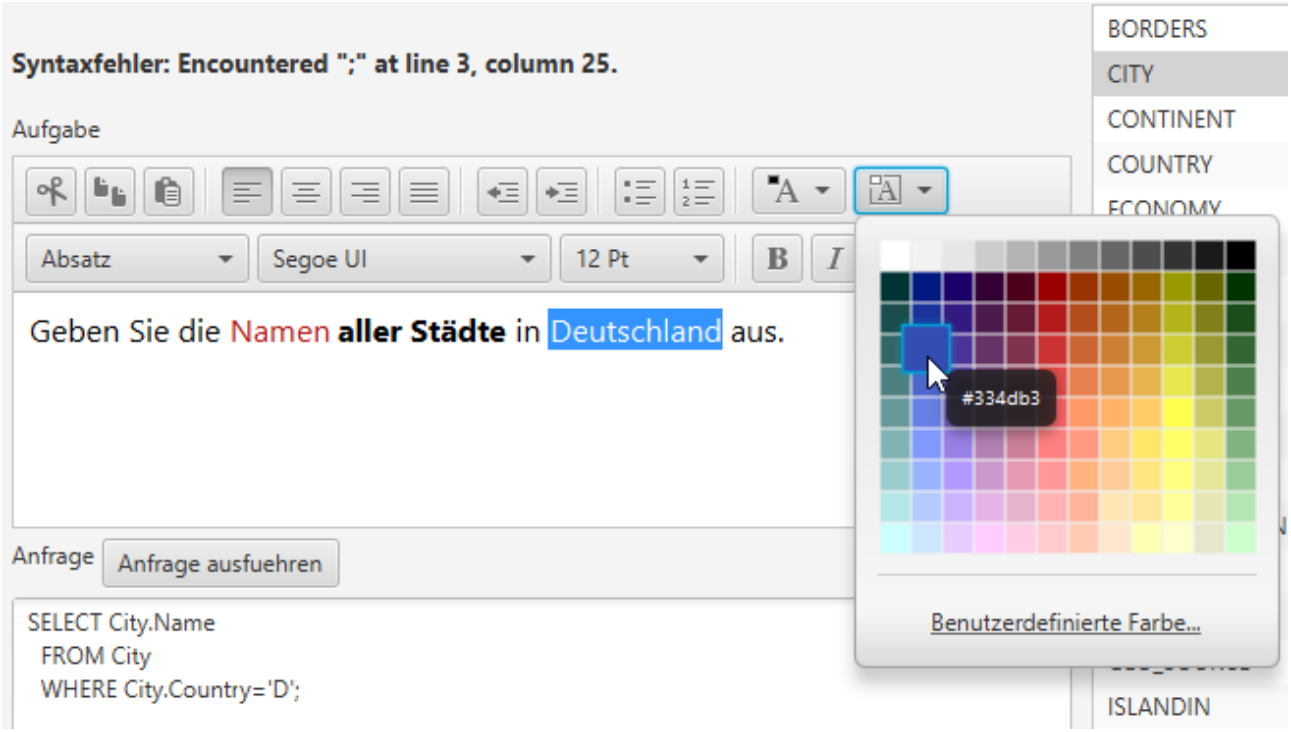
Die Angabe „ok“ und „nok“ gibt an, ob die Anfrage schon in das vom SQLChecker genutzte Format transformiert wurde. Ein „ok“ garantiert nicht, dass die letzte Version übersetzt wurde, hierfür ist der Nutzer selbst verantwortlich.

Soll eine neue Aufgabe erstellt werden, sind alle Felder mit dem Knopf „Neu“ löschbar. Alternativ ist eine vorhandene Anfrage modifizierbar und muss nur einen neuen Dateinamen erhalten.

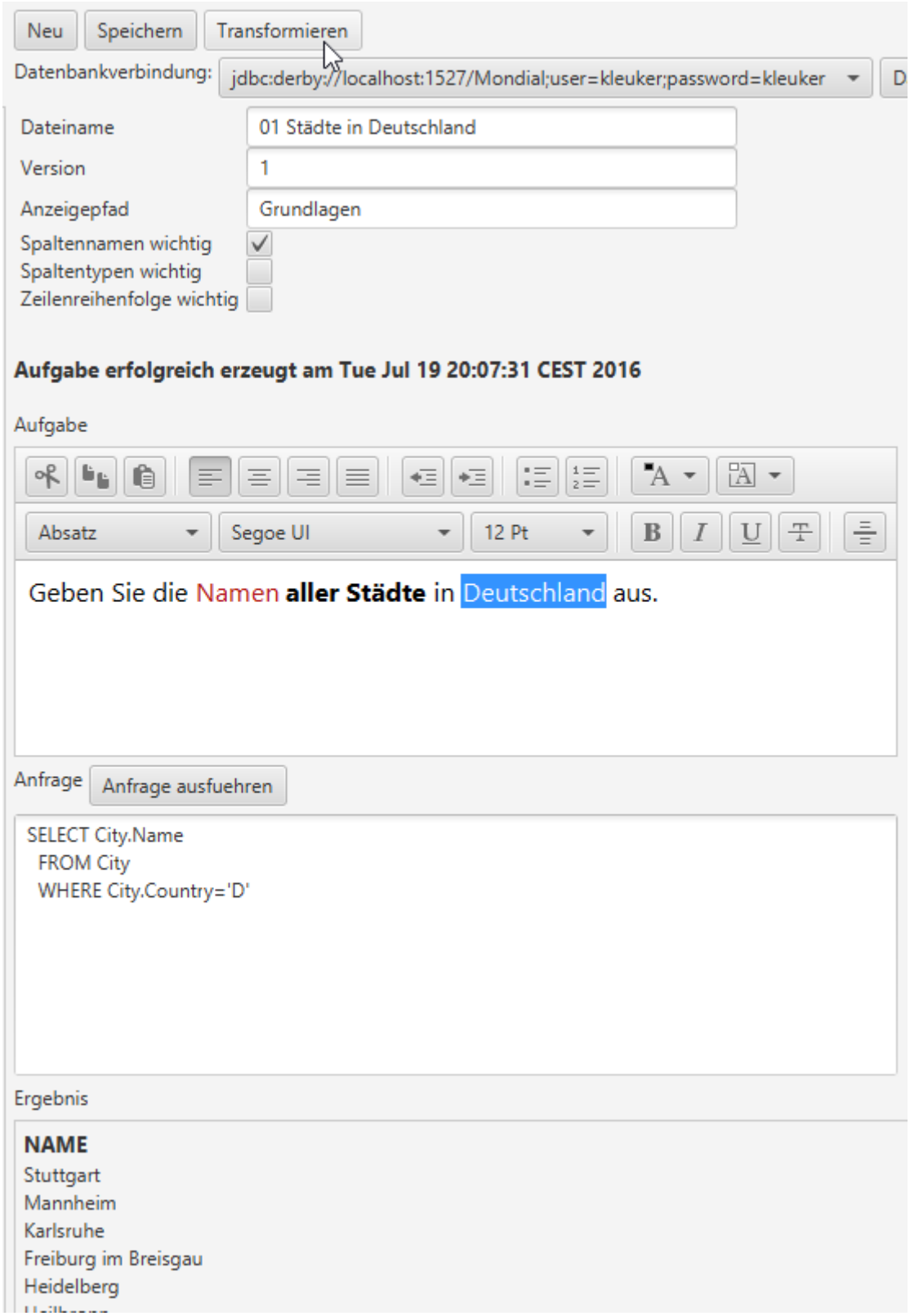


Die eigentliche Aufgabe wird im Editor unter „Aufgabe“ eingegeben. Dieser Editor unterstützt einige graphische Elemente, die optional genutzt werden können. Da es bewusst keine Felder für Autoren oder die genutzte Datenbank gibt, kann solch eine Information auch hier ergänzt werden.

Im Feld Anfrage wird die zugehörige Anfrage eingegeben, die die später erwartete Lösung liefert. Die Anfrage wird mit dem Knopf „Anfrage ausfuehren“ durchgeführt. Über etwaige Fehler wird man im Informationsfeld über der „Aufgabe“ informiert.

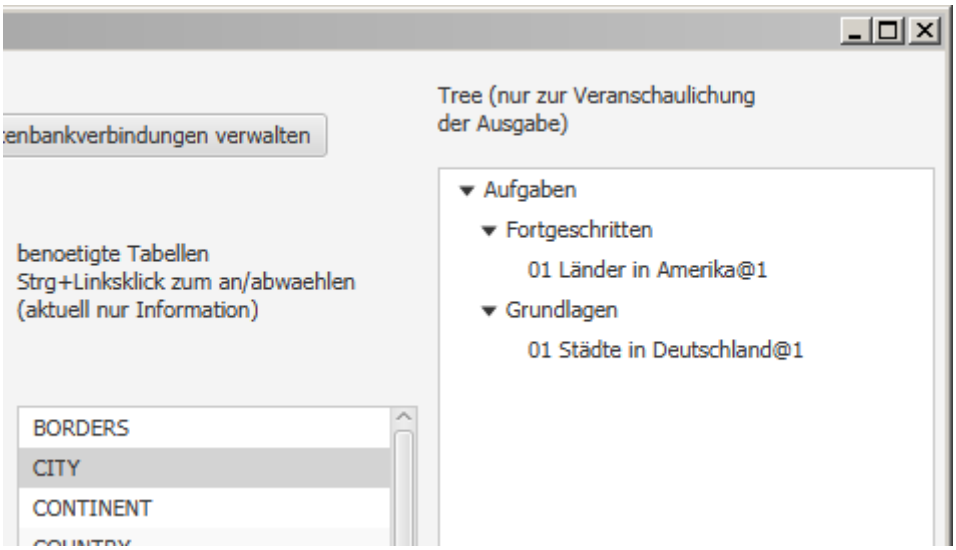


Nach einer erfolgreichen Durchführung der Anfrage wird das zugehörige Ergebnis im unteren Feld ausgegeben. Die Anfrage kann jederzeit mit „Speichern“ gespeichert werden. Erst nach dem Speichern und dem erfolgreichen Ausführen der Anfrage kann die Aufgabe über den Knopf „Transformieren“ in eine Aufgabe für den SQLChecker transformiert werden. Bei Problemen gäbe es einen Hinweis.

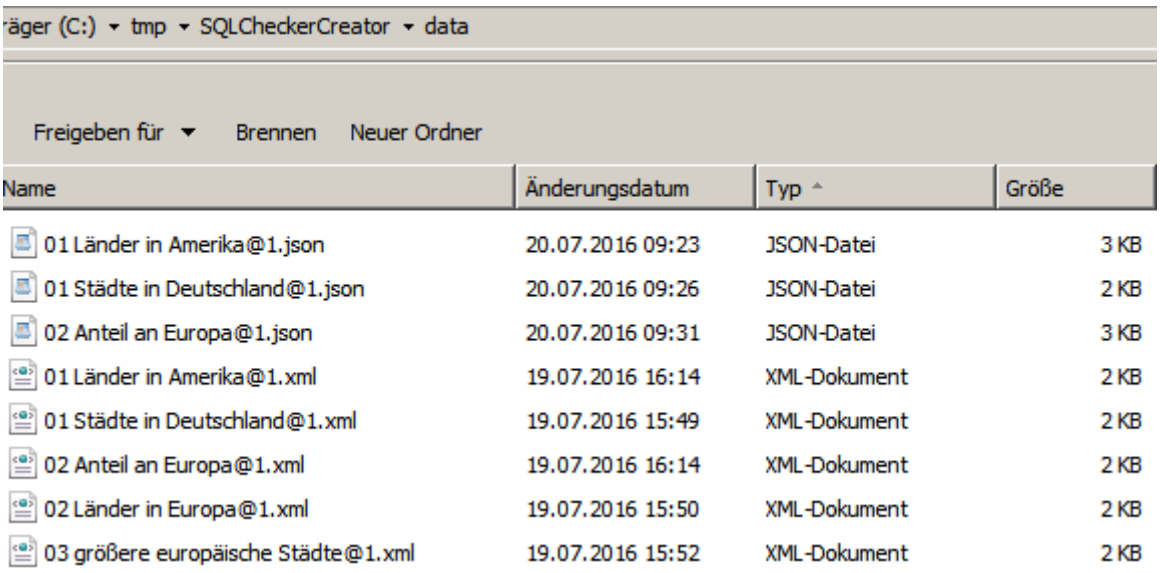

Neu
Speichern
Transformieren
Datenbankverbindung:
jdbc:derby://localhost:1527/Mondial;user=kleuker;password=kleuker
Dateiname
01 Städte in Deutschland
Version
1
Anzeigepfad
Grundlagen
Spaltennamen wichtig
Spaltentypen wichtig
Zeilenreihenfolge wichtig
Aufgabe erfolgreich erzeugt am Tue Jul 19 20:07:31 CEST 2016
Aufgabe
Absatz
Segoe UI
12 Pt
Geben Sie die Namen aller Städte in Deutschland aus.
Anfrage
Anfrage ausfuehren
SELECT City.Name
FROM City
WHERE City.Country='D'
Ergebnis
NAME
Stuttgart
Mannheim
Karlsruhe
Freiburg im Breisgau
Heidelberg

Auf der rechten Seite befindet sich ein Informationsfeld mit den in der aktuellen Datenbankverbindung nutzbaren Tabellen. Diese Tabellen sind selektierbar, werden aber im aktuellen Stand des Werkzeugs nicht weiter genutzt und können ignoriert werden.

Ganz rechts wird der Baum angezeigt, der im SQLChecker mit den vorhandenen Aufgaben ausgegeben wird. Dabei sind nur Aufgaben eingetragen, die transformiert wurden. Im Baum können Aufgaben auch selektiert, aber nicht gelöscht werden.



Um die mit dem SQLCheckerCreator erstellten Aufgaben im SQLChecker zu nutzen, sind die Ergebnisse der Transformation in das Verzeichnis „data“ des SQLCheckers zu kopieren. Die zu kopierenden Aufgaben befinden sich im Verzeichnis „data“ des SQLCheckerCreators. Dateien mit der Endung „.xml“ beinhalten die eigentlichen Aufgaben, Dateien mit der Endung „.json“ die transformierten Aufgaben, die vom SQLChecker genutzt werden können.



Name	Änderungsdatum	Typ	Größe
01 Länder in Amerika@1.json	20.07.2016 09:23	JSON-Datei	3 KB
01 Städte in Deutschland@1.json	20.07.2016 09:26	JSON-Datei	2 KB
02 Anteil an Europa@1.json	20.07.2016 09:31	JSON-Datei	3 KB
01 Länder in Amerika@1.xml	19.07.2016 16:14	XML-Dokument	2 KB
01 Städte in Deutschland@1.xml	19.07.2016 15:49	XML-Dokument	2 KB
02 Anteil an Europa@1.xml	19.07.2016 16:14	XML-Dokument	2 KB
02 Länder in Europa@1.xml	19.07.2016 15:50	XML-Dokument	2 KB
03 größere europäische Städte@1.xml	19.07.2016 15:52	XML-Dokument	2 KB